aws

自訂標籤指南

Rekognition

Rekognition: 自訂標籤指南

Table of Contents

什麼是 Amazon Rekognition 自訂標籤？

使用 Amazon Rekognition 自訂標籤，您可以識別特定於您業務需求的影像中的物件、標誌和場景。例如，您可以在社交媒體帖子中找到您的徽標，在商店貨架上識別您的產品，在裝配線中對機器零件進行分類，區分健康和受感染的植物，或檢測圖像中的動畫人物。

開發自訂模型以分析影像是一項重要的工作，需要時間、專業知識和資源。通常需要幾個月的時間才能完成。此外，它可能需要數千或數萬張手動標籤的影像，才能為模型提供足夠的資料以準確地做出決策。產生這些資料可能需要數月的時間才能收集，而且可能需要大量的貼標員為機器學習做好準備。

亞馬遜 Rekognition 自訂標籤擴展了 Amazon Rekognition 的現有功能，這些功能已經在許多類別的數千萬張圖像上進行了訓練。您可以上傳針對您使用案例的一小組訓練影像 (通常是幾百個或更少)，而不是數千張影像。您可以使用 easy-to-use 主控台執行這項作業。如果您的影像已經加上標籤，Amazon Rekognition 自訂標籤可以在短時間內開始訓練模型。如果沒有，您可以直接在標籤界面中標記圖像，也可以使用 Amazon SageMaker Ground Truth 為您標記圖像。

Amazon Rekognition 自訂標籤從您的映像集開始訓練之後，它可以在短短幾個小時內為您產生自訂影像分析模型。在幕後，Amazon Rekognition 自訂標籤會自動載入和檢查訓練資料、選取正確的機器學習演算法、訓練模型，以及提供模型效能指標。然後，您可以透過 Amazon Rekognition 自訂標籤 API 使用自訂模型，並將其整合到您的應用程式中。

主題

主要好處

簡化資料標籤

Amazon Rekognition 自訂標籤主控台提供視覺化介面，讓您輕鬆快速地標記影像。該界面允許您將標籤應用於整個圖像。您也可以使用具有 click-and-drag 介面的邊界方框，識別和標示影像中的特定物件。或者，如果您擁有大型資料集，則可以使用 Amazon SageMaker Ground Truth 有效地大規模標記映像檔。

Machine Learning

無需機器學習專業知識即可建立您的自訂模型。Amazon Rekognition 自訂標籤包含自動化機器學習 (AutoML) 功能，可為您處理機器學習。提供訓練映像後，Amazon Rekognition 自訂標籤可以自動載入和檢查資料、選取正確的機器學習演算法、訓練模型，以及提供模型效能指標。

簡化模型評估、推論和意見反應

您可以在測試集上評估自訂模型的效能。對於測試集中的每個圖像，您可以看到模型的預測與分配的標籤的 side-by-side 比較。您也可以檢閱詳細的效能指標，例如精確度、召回、F1 分數和可信度分數。您可以立即開始使用模型進行影像分析，或者您可以使用更多影像來重新訓練新版本，以提升效能。開始使用模型後，您可以追蹤預測、更正任何錯誤，並使用意見反應資料重新訓練新模型版本並改善效能。

選擇使用 Amazon Rekognition 自訂標籤

Amazon Rekognition 提供兩種功能，可讓您用來尋找影像中的標籤 (物件、場景和概念)：Amazon Rekognition 自訂標籤和 Amazon Rekognition 影像標籤偵測。請使用下列資訊來決定您應該使用的功能。

Amazon Rekognition 圖像標籤檢測

您可以使用 Amazon Rekognition Image 中的標籤偵測功能，大規模地識別、分類和搜尋影像和視訊中的常見標籤，而無需建立機器學習模型。例如，您可以輕鬆偵測數千種常見物體，例如汽車和卡車、番茄、籃球和足球。

如果您的應用程式需要尋找常見的標籤，我們建議您使用 Amazon Rekognition 影像標籤偵測，因為您不需要訓練模型。若要取得 Amazon Rekognition 影像標籤偵測找到的標籤清單，請參閱偵測標籤。

如果您的應用程式需要尋找 Amazon Rekognition 影像標籤偵測找不到的標籤，例如組裝線上的自訂機器零件，建議您使用 Amazon Rekognition 自訂標籤。

亞馬遜自定義標籤

您可以使用 Amazon Rekognition 自訂標籤輕鬆訓練機器學習模型，在影像中尋找符合您業務需求的特有標籤 (物件、標誌、場景和概念)。

Amazon Rekognition 自訂標籤可以將影像分類 (影像層級預測) 或偵測影像中的物件位置 (物件/邊界框層級預測)。

Amazon Rekognition 自訂標籤為您可偵測的物件和場景類型提供更大的彈性。例如，您可以使用 Amazon Rekognition 影像標籤偵測來尋找植物和樹葉。若要區分健康、受損和受感染的植物，您需要使用 Amazon Rekognition 自訂標籤。

以下是如何使用 Amazon Rekognition 自訂標籤。

- 識別球衣和頭盔上的球隊標誌
- 區分裝配線上的特定機器零件或產品
- 識別媒體庫中的卡通人物
- 在零售貨架上找到特定品牌的產品
- 分類農產品質量（如腐爛，成熟或生的）

> **Note**
>
> Amazon Rekognition 自訂標籤不適用於分析臉孔、偵測文字或尋找影像中不安全的影像內容。若要執行這些任務，您可以使用 Amazon Rekognition 映像。如需詳細資訊，請參閱什麼是 Amazon Rekognition。

您第一次使用 Amazon Rekognition 自訂標籤嗎？

如果您是第一次使用 Amazon Rekognition 自訂標籤，建議您依序閱讀以下章節：

1. 設定 Amazon Rekognition 自訂標籤— 在本節中，您可以設置您的帳戶詳細信息。
2. 了解 Amazon Rekognition 自訂標籤— 在本節中，您將瞭解建立模型的工作流程。
3. 開始使用亞馬遜自訂標籤— 在本節中，您會使用 Amazon Rekognition 自訂標籤建立的範例專案來訓練模型。
4. 教學課程：分類影像— 在本節中，您將學習如何訓練使用您建立的資料集對影像進行分類的模型。

設定 Amazon Rekognition 自訂標籤

以下指示介紹如何設定 Amazon Rekognition 自訂標籤主控台和 SDK。

請注意，您可以透過以下瀏覽器使用 Amazon Rekognition 自訂標籤主控台：

- Chrome - 版本 21 或更新的版本
- Firefox - 版本 27 或更新的版本
- Microsoft Edge - 版本 88 或更新的版本。
- Safari - 版本 7 或更新的版本。此外，您無法使用 Safari 透過 Amazon Rekognition 自訂標籤主控台繪製邊界框。如需詳細資訊，請參閱 使用週框方塊標記物件。

初次使用 Amazon Rekognition 自訂標籤 之前，請先完成以下任務：

主題

步驟 1：建立 AWS 帳戶

在此步驟中，您會建立 AWS 帳戶、建立系統管理使用者，以及了解如何授與 AWS SDK 的程式設計存取權。

主題

註冊 AWS 帳戶

如果您還沒有 AWS 帳戶，請完成以下步驟建立新帳戶。

註冊 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。

   註冊 AWS 帳戶時，會建立 AWS 帳戶根使用者。根使用者有權存取該帳戶中的所有 AWS 服務和資源。作為最佳安全實務，將管理存取權指派給管理使用者，並且僅使用根使用者來執行需要根使用者存取權的任務。

註冊程序完成後，AWS 會傳送一封確認電子郵件給您。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。

建立管理使用者

當您註冊 AWS 帳戶 之後，請保護您的 AWS 帳戶根使用者，啟用 AWS IAM Identity Center，並建立管理使用者，讓您可以不使用根使用者處理日常作業。

保護您的 AWS 帳戶根使用者

1. 選擇 根使用者 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)。

建立管理使用者

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 Identity Center 中，將管理權限授予管理使用者。

若要取得有關使用 IAM Identity Center 目錄 做為身分識別來源的教學課程，請參閱《使用 AWS IAM Identity Center 使用者指南中的以預設 IAM Identity Center 目錄 設定使用者存取權限。

以管理員的身分登入

- 若要使用您的 IAM 身分中心使用者登入，請使用建立 IAM 身分中心使用者時傳送至您電子郵件地址的登入 URL。

  如需有關如何使用 IAM Identity Center 使用者登入的說明，請參閱《AWS 登入 使用者指南》中的登入 AWS存取入口網站。

程式設計存取權

若使用者想要與 AWS Management Console 之外的 AWS 互動，則需要程式設計存取權。授予程式設計存取權的方式取決於存取 AWS 的使用者類型。

若要授予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)	使用臨時憑證簽署對 AWS CLI、AWS SDKs 或 AWS APIs 的程式設計請求。	請依照您要使用的介面所提供的指示操作。 • 關於 AWS CLI，請參閱 AWS Command Line Interface使用者指南 中的 設定 AWS CLI 來使用 AWS IAM Identity Center。 • 關於 AWS SDKs、工具和 AWS APIs，請參閱 AWSSDKs 和工具參考指南 中的 IAM Identity Center 驗證。

哪個使用者需要程式設計存取權？	到	By
IAM	使用臨時憑證簽署對 AWS CLI、AWS SDKs 或 AWS APIs 的程式設計請求。	請遵循 IAM 使用者指南 中 使用臨時憑證搭配 AWS 資源 中的指示。
IAM	(不建議使用) 使用長期憑證簽署 AWS CLI、AWS SDKs 或 AWS APIs 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。 • 關於 AWS CLI，請參閱 AWS Command Line Interface使用者指南 中的 使用 IAM 使用者憑證進行驗證。 • 關於 AWS SDKs 和工具，請參閱 AWSSDKs 和工具參考指南 中的 使用長期憑證進行驗證。 • 關於 AWS API，請參閱 IAM 使用者指南 中的 管理 IAM 使用者的存取金鑰。

步驟 2：設定 Amazon Rekognition 自訂標籤主控台權限

若要使用 Amazon Rekognition 主控台，您必須新增才能擁有適當的權限。如果您想要將培訓檔案儲存在 主控台儲存貯體 中，則需要額外的權限。

主題

- 允許主控台存取
- 存取外部 Amazon S3 儲存貯體
- 指派權限

允許主控台存取

若要使用 Amazon Rekognition 自訂標籤主控台，您需要下列涵蓋 Amazon S3、G SageMaker round Truth 和 Amazon Rekognition 自訂標籤的 IAM 政策。如需關於指派權限的更多詳細資訊，請參閱 指派權限。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:ListBucket",
                "s3:ListAllMyBuckets"
            ],
            "Resource": "*"
        },
        {
            "Sid": "s3Policies",
            "Effect": "Allow",
            "Action": [
                "s3:ListBucket",
                "s3:CreateBucket",
                "s3:GetBucketAcl",
                "s3:GetBucketLocation",
                "s3:GetObject",
                "s3:GetObjectAcl",
                "s3:GetObjectVersion",
                "s3:GetObjectTagging",
                "s3:GetBucketVersioning",
                "s3:GetObjectVersionTagging",
                "s3:PutBucketCORS",
                "s3:PutLifecycleConfiguration",
                "s3:PutBucketPolicy",
                "s3:PutObject",
                "s3:PutObjectTagging",
                "s3:PutBucketVersioning",
                "s3:PutObjectVersionTagging"
            ],
            "Resource": [
                "arn:aws:s3:::custom-labels-console-*"

            ]
```

```
        },
        {
            "Sid": "rekognitionPolicies",
            "Effect": "Allow",
            "Action": [
                "rekognition:*"
            ],
            "Resource": "*"
        },
        {
            "Sid": "groundTruthPolicies",
            "Effect": "Allow",
            "Action": [
                "groundtruthlabeling:*"
            ],
            "Resource": "*"
        }
    ]
}
```

存取外部 Amazon S3 儲存貯體

當您第一次在新 AWS 區域中開啟 Amazon Rekognition 自訂標籤主控台時，Amazon Rekognition 自訂標籤會建立用於存放專案檔案的儲存貯體 (主控台儲存貯體)。或者，您可以使用自己的 Amazon S3 儲存貯體 (外部儲存貯體) 將圖像或清單檔案上傳到主控台。若要使用外部儲存貯體，請將下列政策區塊新增至前述策略。用您的儲存貯體名稱取代 `my-bucket`。

```
        {
            "Sid": "s3ExternalBucketPolicies",
            "Effect": "Allow",
            "Action": [
                "s3:GetBucketAcl",
                "s3:GetBucketLocation",
                "s3:GetObject",
                "s3:GetObjectAcl",
                "s3:GetObjectVersion",
                "s3:GetObjectTagging",
                "s3:ListBucket",
                "s3:PutObject"
            ],
            "Resource": [
                "arn:aws:s3:::my-bucket*"
            ]
```

```
        }
```

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：

- AWS IAM Identity Center 中的使用者和群組：

  建立權限合集。請遵循 AWS IAM Identity Center 使用者指南 的 建立權限合集 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

您可以使用 Amazon Rekognition 自訂標籤專案建立和管理模型。當您第一次在新 AWS 區域中開啟 Amazon Rekognition 自訂標籤主控台時，Amazon Rekognition 自訂標籤會建立 Amazon S3 儲存貯體 (主控台儲存貯體) 來存放您的專案。您應該記下主控台儲存貯體的名稱，以便稍後引用，因為您可能需要在 AWS SDK 操作或主控台任務（例如建立資料集）中使用該儲存貯體名稱。

儲存貯體名稱的格式為 `custom-labels-console-<##>-###`。隨機值可確保儲存貯體名稱之間不會發生衝突。

若要建立主控台儲存貯體

1. 確定使用者擁有正確的權限。如需詳細資訊，請參閱 允許主控台存取。
2. 登入 AWS Management Console，並在 https://console.aws.amazon.com/rekognition/ 開啟 Amazon Rekognition 主控台。
3. 選擇 開始使用。
4. 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 對話框中執行以下操作：

   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 Rekognition 自訂標籤代表您建立 Amazon S3 儲存貯體 (主控台儲存貯體)。

5. 關閉瀏覽器視窗。

步驟 4：設定 AWS CLI 和 AWS SDKs

您可以搭配 AWS Command Line Interface (AWS CLI) 和 AWS SDKs 使用 Amazon Rekognition 自訂標籤。如果您需要從終端執行 Amazon Rekognition 自訂標籤操作，請安裝 AWS CLI。如果您要建立應用程式，請下載所使用之程式設計語言的 AWS SDK。

主題

- 安裝 AWS SDKS
- 授予程式設計存取權
- 設定 SDK 權限
- 呼叫 Amazon Rekognition 自訂標籤操作

安裝 AWS SDKS

請遵循以下步驟來下載及設定 AWS SDK。

設定 AWS CLI 和 AWS SDKs

- 下載並安裝您要使用的 AWS CLI 和 AWS SDKs。本指南提供了 AWS CLI、Java 和 Python 的範例。如需關於 AWS SDKs 的更多詳細資訊，請參閱 適用於 Amazon Web Services 的工具。

授予程式設計存取權

您可以在本機電腦或其他 AWS 環境 (例如 Amazon Elastic Compute Cloud 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 操作的存取權。

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，建議您使用短期憑證授予使用者存取 AWS SDK 操作。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。

若使用者想要與 AWS Management Console 之外的 AWS 互動，則需要程式設計存取權。授予程式設計存取權的方式取決於存取 AWS 的使用者類型。

若要授予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)	使用臨時憑證簽署對 AWS CLI、AWS SDKs 或 AWS APIs 的程式設計請求。	請依照您要使用的介面所提供的指示操作。 • 關於 AWS CLI，請參閱 AWS Command Line Interface使用者指南 中的 設定 AWS CLI 來使用 AWS IAM Identity Center。 • 關於 AWS SDKs、工具和 AWS APIs，請參閱 AWSSDKs 和工具參考指南 中的 IAM Identity Center 驗證。
IAM	使用臨時憑證簽署對 AWS CLI、AWS SDKs 或 AWS APIs 的程式設計請求。	請遵循 IAM 使用者指南 中 使用臨時憑證搭配 AWS 資源 中的指示。
IAM	(不建議使用) 使用長期憑證簽署 AWS CLI、AWS SDKs 或 AWS APIs 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。 • 關於 AWS CLI，請參閱 AWS Command Line Interface使用者指南 中的 使用 IAM 使用者憑證進行驗證。

哪個使用者需要程式設計存取權？	到	By
		• 關於 AWS SDKs 和工具，請參閱 AWSSDKs 和工具參考指南 中的 使用長期憑證進行驗證。 • 關於 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 在本機電腦執行程式碼 中建立的短期憑證來執行本指南中的 AWS CLI 和程式碼範例。若要取得憑證和其他設定資訊，範例會使用名為 `custom-labels-access` 的設定檔，例如：

```
session = boto3.Session(profile_name='custom-labels-access')
rekognition_client = session.client("rekognition")
```

設定檔所代表的使用者必須具有呼叫 Amazon Rekognition 自訂標籤 SDK 操作和範例所需的其他 AWS SDK 操作的權限。如需詳細資訊，請參閱 設定 SDK 權限。如要指派權限，請參閱 設定 SDK 權限。

若要建立適用於 AWS CLI 和程式碼範例使用的設定檔，請選擇下列其中一項。請確定您建立的設定檔名稱是 `custom-labels-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。
- 人力身分 (由 AWS IAM Identity Center 管理的使用者) — 按照 配置 AWS CLI 以使用 AWS IAM Identity Center 的指示進行操作。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證在 AWS 環境中簽署 AWS SDK 呼叫，例如在 AWS Lambda 函數中運行的生產程式碼。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憑證。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。

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 Rekognition 自訂標籤 SDK 操作，您需要存取 Amazon Rekognition 自訂標籤 API 和用於模型培訓的 Amazon S3 儲存貯體的權限。

主題

- 授予 SDK 操作權限
- 使用 AWS SDK 的政策更新
- 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 DetectCustomLabels，您需要執行許可rekognition:DetectCustomLabels。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。您可以使用 AmazonRekognitionCustomLabelsFullAccess AWS 受管政策取得對 Amazon Rekognition 自訂標籤 API 的完整存取權。如需詳細資訊，請參閱 AWS 受管政策： AmazonRekognitionCustomLabelsFullAccess。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。

如要指派權限,請參閱 指派權限。

使用 AWS SDK 的政策更新

若要將 AWS SDK 與最新版本的 Amazon Rekognition 自訂標籤結合使用，您不再需要授予 Amazon Rekognition 自訂標籤權限，即可存取包含您的培訓和測試圖像的 Amazon S3 儲存貯體。如果您之前已新增權限，您不需要移除這些權限。如果您選擇這樣做，請從主體服務為 rekognition.amazonaws.com 的儲存貯體中刪除任何政策。例如：

```
"Principal": {
    "Service": "rekognition.amazonaws.com"
}
```

如需更多詳細資訊，請參閱 使用儲存貯體政策。

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：

- AWS IAM Identity Center 中的使用者和群組：

  建立權限合集。請遵循 AWS IAM Identity Center 使用者指南 的 建立權限合集 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 新增權限至使用者 (主控台) 中的指示。

呼叫 Amazon Rekognition 自訂標籤操作

執行以下程式碼，以確認您可以呼叫 Amazon Rekognition 自訂標籤 API。此程式碼會在目前 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 `DescribeProjects`操作。

一般而言，呼叫範例函數需要 AWS SDK Rekognition 用戶端和任何其他必要的參數。AWS SDK 用戶端在主函數中宣告。

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。另外請檢查您用作 Amazon Rekognition 自訂標籤的 AWS 區域是否在所有 AWS 區域都可用。

若要呼叫 Amazon Rekognition 自訂標籤操作

1. 如果您執行此操作 ，請先安裝並配置 AWS CLI 和 AWS SDKs。如需詳細資訊，請參閱 步驟 4：設定 AWS CLI 和 AWS SDKs。
2. 使用以下的範例程式碼來檢視您的專案。

   CLI

   使用 `describe-projects` 指令列出您帳戶中的專案。

   ```
   aws rekognition describe-projects \
   --profile custom-labels-access
   ```

   Python

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

   """
   This example shows how to describe your Amazon Rekognition Custom Labels
    projects.
   If you haven't previously created a project in the current AWS Region,
   the response is an empty list, but does confirm that you can call an
   ```

```
Amazon Rekognition Custom Labels operation.
"""
from botocore.exceptions import ClientError
import boto3

def describe_projects(rekognition_client):
    """
    Lists information about the projects that are in in your AWS account
    and in the current AWS Region.

    : param rekognition_client: A Boto3 Rekognition client.
    """
    try:
        response = rekognition_client.describe_projects()
        for project in response["ProjectDescriptions"]:
            print("Status: " + project["Status"])
            print("ARN: " + project["ProjectArn"])
            print()
        print("Done!")
    except ClientError as err:
        print(f"Couldn't describe projects. \n{err}")
        raise


def main():
    """
    Entrypoint for script.
    """

    session = boto3.Session(profile_name='custom-labels-access')
    rekognition_client = session.client("rekognition")

    describe_projects(rekognition_client)


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

Java V2

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

```
*/

package com.example.rekognition;

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

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.DatasetMetadata;
import
 software.amazon.awssdk.services.rekognition.model.DescribeProjectsRequest;
import
 software.amazon.awssdk.services.rekognition.model.DescribeProjectsResponse;
import software.amazon.awssdk.services.rekognition.model.ProjectDescription;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;

public class Hello {

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

    public static void describeMyProjects(RekognitionClient rekClient) {

        DescribeProjectsRequest descProjects = null;

        // If a single project name is supplied, build projectNames argument

        List<String> projectNames = new ArrayList<String>();


        descProjects = DescribeProjectsRequest.builder().build();

        // Display useful information for each project.

        DescribeProjectsResponse resp =
 rekClient.describeProjects(descProjects);

        for (ProjectDescription projectDescription : resp.projectDescriptions())
 {

            System.out.println("ARN: " + projectDescription.projectArn());
```

```
            System.out.println("Status: " +
 projectDescription.statusAsString());
            if (projectDescription.hasDatasets()) {
                for (DatasetMetadata datasetDescription :
 projectDescription.datasets()) {
                    System.out.println("\tdataset Type: " +
 datasetDescription.datasetTypeAsString());
                    System.out.println("\tdataset ARN: " +
 datasetDescription.datasetArn());
                    System.out.println("\tdataset Status: " +
 datasetDescription.statusAsString());
                }
            }
            System.out.println();
        }

    }

    public static void main(String[] args) {

        try {

            // Get the Rekognition client
            RekognitionClient rekClient = RekognitionClient.builder()
                .credentialsProvider(ProfileCredentialsProvider.create("custom-
labels-access"))
                .region(Region.US_WEST_2)
                .build();

            // Describe projects

            describeMyProjects(rekClient);

            rekClient.close();

        } catch (RekognitionException rekError) {
            logger.log(Level.SEVERE, "Rekognition client error: {0}",
 rekError.getMessage());
            System.exit(1);
        }

    }
```

```
}
```

步驟 5：(可選) 加密培訓檔案

您可以選擇以下其中一個選項來加密 Amazon Rekognition 自訂標籤位於主控台儲存貯體或外部 Amazon S3 儲存貯體中的清單檔案和圖像檔。

- 使用 Amazon S3 金鑰 (SSE-S3)。
- 使用您的 AWS KMS key。

> **Note**
>
> 呼叫 IAM 主體 需要權限才能解密檔案。如需詳細資訊，請參閱 解密使用 AWS Key Management Service 加密的文件。

如需更多加密 Amazon S3 儲存貯體的詳細資訊，請參閱 設定 Amazon S3 儲存貯體的預設伺服器端加密行為。

解密使用 AWS Key Management Service 加密的文件

如果您使用 AWS Key Management Service (KMS) 加密 Amazon Rekognition 自訂標籤清單檔案和圖像檔案，請將呼叫 Amazon Rekognition 自訂標籤的 IAM 主體新增至 KMS 金鑰的金鑰政策。這樣做可讓 Amazon Rekognition 自訂標籤在培訓前解密您的清單檔案和圖像檔案。如需更多詳細資訊，請參閱 我的 Amazon S3 儲存貯體使用自訂 AWS KMS 金鑰預設加密。如何允許使用者從儲存貯體中下載及上傳？

IAM 主體需要 KMS 金鑰的以下權限。

- 公里：GenerateDataKey
- kms:解密

如需更多詳細資訊，請參閱 使用儲存在 AWS Key Management Service (SSE-KMS) 中的 KMS 金鑰進行伺服器端加密來保護資料。

加密複製的培訓和測試圖像

為了培訓您的模型，Amazon Rekognition 自訂標籤會製作一份來源培訓和測試圖像的複本。在預設情況下，複製的圖像會使用 AWS 擁有和管理的金鑰進行靜態加密。您也可以選擇使用自己的 AWS KMS key。如果您使用自己的 KMS 金鑰，您需要 KMS 金鑰的以下權限。

- 公里：CreateGrant
- 公里：DescribeKey

當您使用主控台培訓模型或呼叫 `CreateProjectVersion` 操作時，您可以選擇性地指定 KMS 金鑰。您使用的 KMS 金鑰不需要與您在 Amazon S3 儲存貯體中用來加密清單檔案和圖像檔案的 KMS 金鑰相同。如需詳細資訊，請參閱 步驟 5：(可選) 加密培訓檔案。

如需更多詳細資訊，請參閱 AWS Key Management Service 概念。您的來源圖像不受影響。

如需培訓模型的資訊，請參閱 訓練 Amazon Rekognition 自訂標籤模型。

步驟 6：(可選) 將舊資料集與新專案建立關聯

Amazon Rekognition 自訂標籤現在可透過專案管理資料集。您建立的早期（舊）資料集是唯讀檔，必須先與專案建立關聯才能使用它們。當您使用主控台開啟專案的詳細資料頁面時，我們會自動將訓培訓專案模型最新版本的資料集與專案建立關聯。如果您使用 AWS SDK，則不會自動將資料集與專案建立關聯。

未關聯的舊資料集從未用於模型培訓，或者用於培訓舊版本的模型。「舊資料集」頁面會顯示所有關聯和未關聯的資料集。

若要使用未經關聯的舊資料集，請在舊資料集的頁面上建立新專案。資料集會成為新專案的培訓資料集。您也可以為已關聯的資料集建立專案，因為舊資料集可以有多個關聯。

將舊資料集與新專案建立關聯

1. 開啟 Amazon Rekognition 主控台：https://console.aws.amazon.com/rekognition/。
2. 在左側視窗中，選擇使用 自訂標籤。Amazon Rekognition 自訂標籤的登入頁面會隨即顯示。
3. 在左側導覽視窗中，選擇 舊資料集。
4. 在資料集檢視中，選擇您要與專案建立關聯的舊資料集。
5. 選擇 使用資料集建立專案。
6. 在 建立專案 的頁面中，在 專案名稱 中輸入新專案的名稱。

7. 選擇 建立專案 以建立專案。專案可能需要一段時間才能建立。
8. 使用專案。如需詳細資訊，請參閱 了解 Amazon Rekognition 自訂標籤。

使用舊資料集作為測試資料集

您可以先將舊資料集與新專案建立關聯，使用舊資料集作為現有專案的測試資料集。然後，您可以將新專案的培訓資料集複製到現有專案的測試資料集。

使用舊資料集作為測試資料集

1. 請跟隨中 步驟 6：(可選) 將舊資料集與新專案建立關聯 的指示，將舊資料集與新專案建立關聯。
2. 使用從新專案複製的訓練資料集，在現有專案中建立測試資料集。如需詳細資訊，請參閱 現有的資料集。
3. 按照 刪除 Amazon Rekognition 自訂標籤專案 (主控台) 中的說明刪除新專案。

或者，您也可以使用舊資料集的清單檔案來建立測試資料集。如需更多詳細資訊，請參閱 建立清單檔案。

了解 Amazon Rekognition 自訂標籤

本節提供工作流程概觀，以便在主控台和AWS SDK 中訓練和使用 Amazon Rekognition 自訂標籤模型。

> **Note**
>
> Amazon Rekognition 自訂標籤現在可以管理專案內的資料集。您可以使用控制台和AWS SDK 為您的項目創建數據集。如果您之前曾使用過 Amazon Rekognition 自訂標籤，您的舊資料集可能需要與新專案建立關聯。如需詳細資訊，請參閱 步驟 6：(可選) 將舊資料集與新專案建立關聯

主題

決定您的型號類型

您首先決定要訓練的模型類型，這取決於您的業務目標。例如，您可以訓練模型以在社交媒體貼文中尋找您的標誌、在商店貨架上識別您的產品，或在裝配線中對機器零件進行分類。

Amazon Rekognition 自訂標籤可訓練以下類型的模型：

- 尋找物件、場景和概念
- 尋找物件位置
- 尋找品牌的位置

為了協助您決定要訓練的模型類型，Amazon Rekognition 自訂標籤提供您可以使用的範例專案。如需詳細資訊，請參閱開始使用亞馬遜自訂標籤。

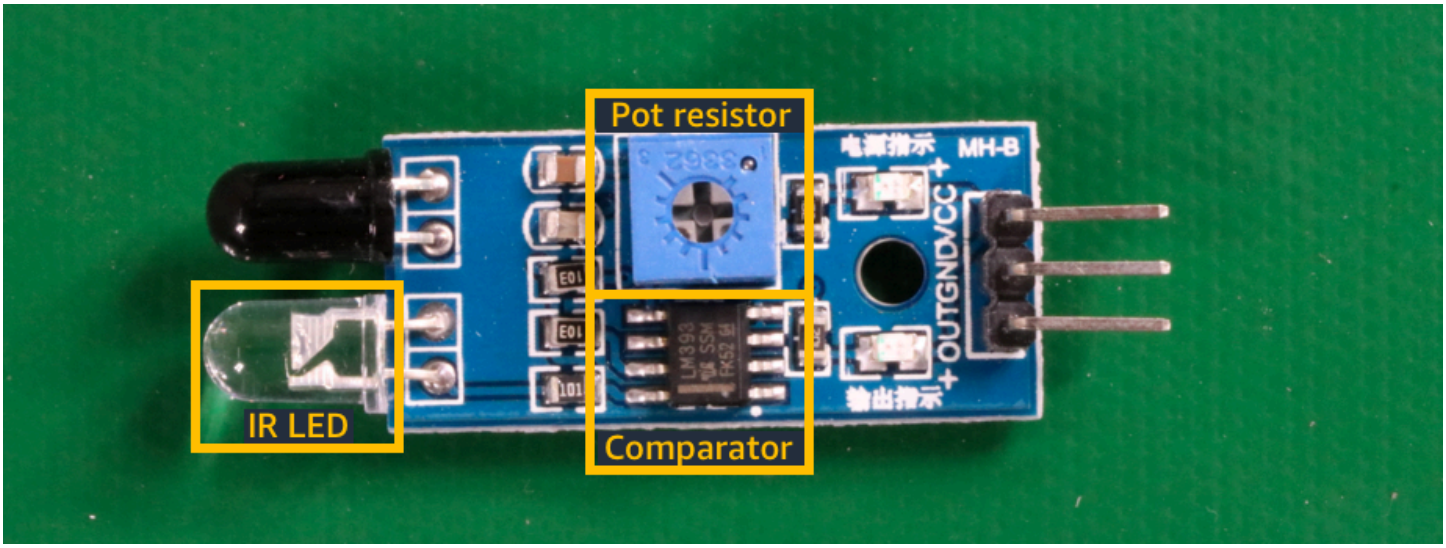
尋找物件、場景和概念

該模型預測與整個圖像相關聯的對象，場景和概念的分類。例如，您可以訓練確定影像是否包含旅遊景點的模型。如需範例專案，請參閱影像分類。

或者，您可以訓練將圖像分類為多個類別的模型。例如，上一個影像可能具有諸如天空顏色、反射或湖泊之類的品類。如需範例專案，請參閱多標籤圖像分類。

尋找物件位置

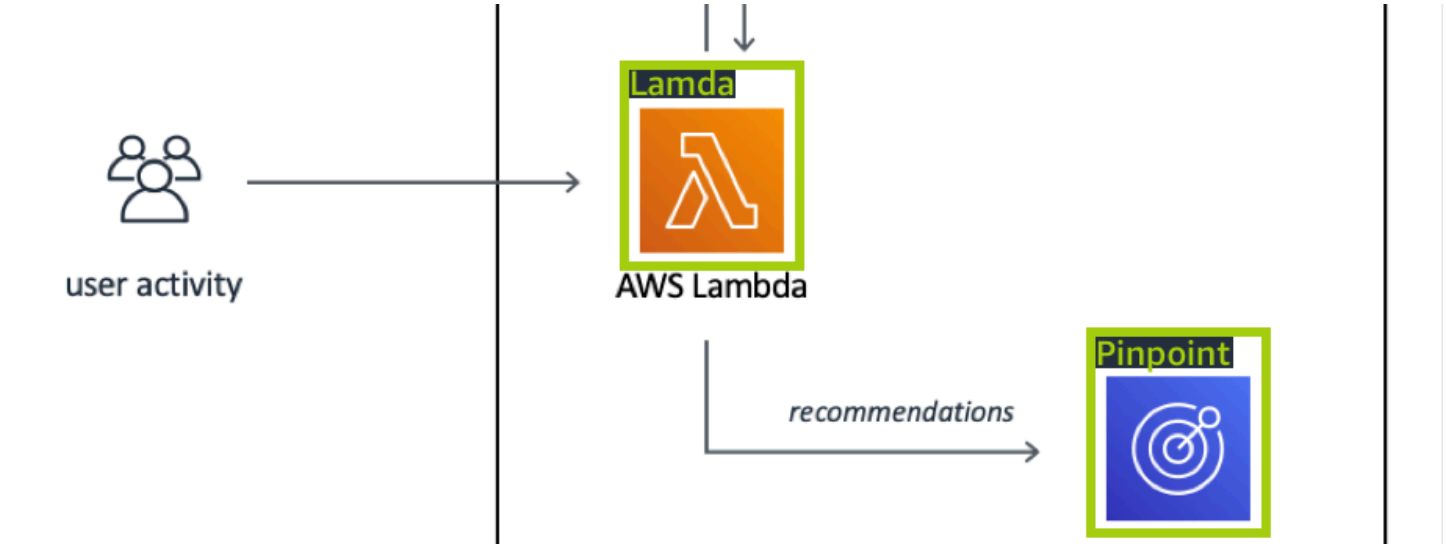
該模型預測圖像上的對象的位置。預測包括物件位置的邊界方框資訊，以及用來識別邊界框內物件的標籤。例如，下圖顯示了電路板各個部分周圍的邊界框，例如比較器或電阻器。



物件本地化範例專案顯示 Amazon Rekognition 自訂標籤如何使用標記的邊界框來訓練尋找物件位置的模型。

尋找品牌的位置

Amazon Rekognition 自訂標籤可訓練在影像上尋找品牌位置 (例如標誌) 的模型。預測包括品牌位置的邊界方框資訊，以及用來識別邊界方框內物件的標籤。如需範例專案，請參閱品牌檢測。



建立模型

建立模型的步驟包括建立專案、建立訓練和測試資料集，以及訓練模型。

建立專案

Amazon Rekognition 自訂標籤專案為建立和管理模型所需的資源群組。專案管理下列項目：

- 資料集 — 用來訓練模型的影像和影像標籤。專案具有訓練資料集和測試資料集。
- 模型 — 您訓練的軟件，以找到您的業務獨特的概念，場景和對象。您可以在專案中擁有多個模型版本。

建議您將專案用於單一使用案例，例如在電路板上尋找電路板零件。

您可以使用 Amazon Rekognition 自訂標籤主控台和 CreateProjectAPI 建立專案。如需詳細資訊，請參閱建立專案。

建立訓練和測試資料集

資料集是描述這些影像的一組影像和標籤。在專案中，您可以建立訓練資料集和測試資料集，Amazon Rekognition 自訂標籤用來訓練和測試模型。

標籤可識別影像中物件周圍的物件、場景、概念或邊界方框。標籤會指定給整個影像 (影像層級)，或是指定給圍繞影像物件的邊界方框。

> **Important**
>
> 如何標記資料集中的影像，會決定 Amazon Rekognition 自訂標籤所建立的模型類型。例如，若要訓練尋找物件、場景和概念的模型，您可以為訓練和測試資料集中的影像指派影像層級標籤。如需詳細資訊，請參閱規劃資料集。

圖像必須為 PNG 和 JPEG 格式，並且您應該遵循輸入的圖像建議。如需詳細資訊，請參閱準備影像。

建立訓練和測試資料集 (主控台)

您可以使用單一資料集，或使用個別的訓練和測試資料集來啟動專案。如果您從單一資料集開始，Amazon Rekognition 自訂標籤會在訓練期間分割您的資料集，以便為您的專案建立訓練資料集 (80%) 和測試資料集 (20%)。如果您希望 Amazon Rekognition 自訂標籤決定要用於訓練和測試的映像，請從單一資料集開始。為了完全控制訓練、測試和效能調整，我們建議您使用個別的訓練和測試資料集來啟動專案。

建立專案的資料集，您可以使用下列其中一種方式匯入影像：

- 從本機電腦匯入影像。
- 從 S3 儲存貯體中匯入映像。Amazon Rekognition 自訂標籤可以使用包含影像的資料夾名稱來標記影像。
- 導入亞馬遜 SageMaker Ground Truth 清單文件。
- 複製現有 Amazon Rekognition 自訂標籤資料集。

如需詳細資訊，請參閱建立包含影像的訓練和測試資料集。

視您匯入影像的位置而定，您的影像可能沒有標記。例如，從本機電腦匯入的影像不會加上標籤。從亞馬遜 SageMaker Ground Truth 清單文件導入的圖像被標記。您可以使用 Amazon Rekognition 自訂標籤主控台來新增、變更和指派標籤。如需詳細資訊，請參閱標記檔案。

若要使用主控台建立訓練和測試資料集，請參閱建立包含影像的訓練和測試資料集。如需包含建立訓練和測試資料集的自學課程，請參閱〈〉教學課程：分類影像。

建立訓練和測試資料集 (SDK)

若要建立訓練和測試資料集，請使用`CreateDataset` API。您可以使用 Amazon Sageemaker 格式資訊清單檔案或複製現有的 Amazon Rekognition 自訂標籤資料集來建立資料集。如需詳細資訊，請參閱建立訓練和測試資料集 (SDK) 如有必要，您可以建立自己的資訊清單檔案。如需詳細資訊，請參閱the section called “建立清單檔案”。

訓練您的模型

使用訓練資料集訓練您的模型。每次訓練模型時，都會建立新版本的模型。在訓練期間，Amazon Rekognition 自訂標籤會測試訓練模型的效能。您可以使用結果來評估和改善模型。訓練需要一段時間才能完成。您只需為成功的模型訓練付費。如需詳細資訊，請參閱訓練 Amazon Rekognition 自訂標籤模型。如果模型訓練失敗，Amazon Rekognition 自訂標籤會提供您可以使用的偵錯資訊。如需詳細資訊，請參閱偵錯失敗的模型訓練。

訓練您的模型（控制台）

若要使用主機訓練模型，請參閱訓練模型（控制台）。

訓練模型 (SDK)

您可以通過調用訓練 Amazon Rekognition 自訂標籤模型CreateProjectVersion。如需詳細資訊，請參閱訓練模型 (SDK)。

改善您的模型

在測試期間，Amazon Rekognition 自訂標籤會建立評估指標，讓您可以使用這些指標來改善訓練過的模型。

評估模型

使用在測試期間建立的效能指標來評估模型的效能。效能指標 (例如 F1、精確度和召回) 可讓您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。如需詳細資訊，請參閱評估模型的指標。

評估模型 (控制台)

檢視效能指標，請參閱存取評估指標 (主控台)。

評估模型 (SDK)

要獲取性能指標，DescribeProjectVersions請致電以獲取測試結果。如需詳細資訊，請參閱存取 Amazon Rekognition 自訂標籤評估指標 (SDK)。測試結果包括控制台中不可用的指標，例如分類結果的混淆矩陣。測試結果以下列格式傳回：

- F1 分數 — 代表模型精確度和召回整體效能的單一值。如需詳細資訊，請參閱F1。
- 摘要檔案位置 — 測試摘要包括整個測試資料集的彙總評估指標，以及每個個別標籤的指標。`DescribeProjectVersions`傳回摘要檔案的 S3 儲存貯體和資料夾位置。如需詳細資訊，請參閱摘要檔案。
- 評估資訊清單快照位置 — 快照集包含有關測試結果的詳細資料，包括信賴度分級和二進位分類測試的結果，例如誤判。 `DescribeProjectVersions`傳回快照檔案的 S3 儲存貯體和資料夾位置。如需詳細資訊，請參閱評估資訊清單。

改善您的模型

如果需要改進，您可以新增更多訓練影像或改善資料集標籤。如需詳細資訊，請參閱改善 Amazon Rekognition 自訂標籤模型。您也可以針對模型所做的預測提供意見反應，並使用它來改善模型。如需詳細資訊，請參閱模型回饋方案。

改善您的模型 (控制台)

若要將影像新增至資料集，請參閱將更多影像新增至資料集。若要加入或變更標示，請參閱the section called “標記檔案”。

若要重新訓練模型，請參閱訓練模型（控制台）。

改善您的模型 (SDK)

若要將影像新增至資料集或變更影像的標籤，請使用UpdateDatasetEntries API。UpdateDatasetEntries更新或將 JSON 行添加到清單文件中。每個 JSON 行都包含單一影像的資訊，例如指派的標籤或邊界方框資訊。如需詳細資訊，請參閱新增更多影像 (SDK)。若要檢視資料集中的項目，請使用ListDatasetEntries API。

若要重新訓練模型，請參閱訓練模型 (SDK)。

啟動模型

在您可以使用模型之前，請先使用 Amazon Rekognition 自訂標籤主控台或StartProjectVersion API 啟動模型。您需支付模型執行個體的執行個體。如需詳細資訊，請參閱執行培訓過的模型。

啟動 (主控台)

若要使用主控台啟動模型，請參閱啟動 Amazon Rekognition 自訂標籤模型 (主控台)。

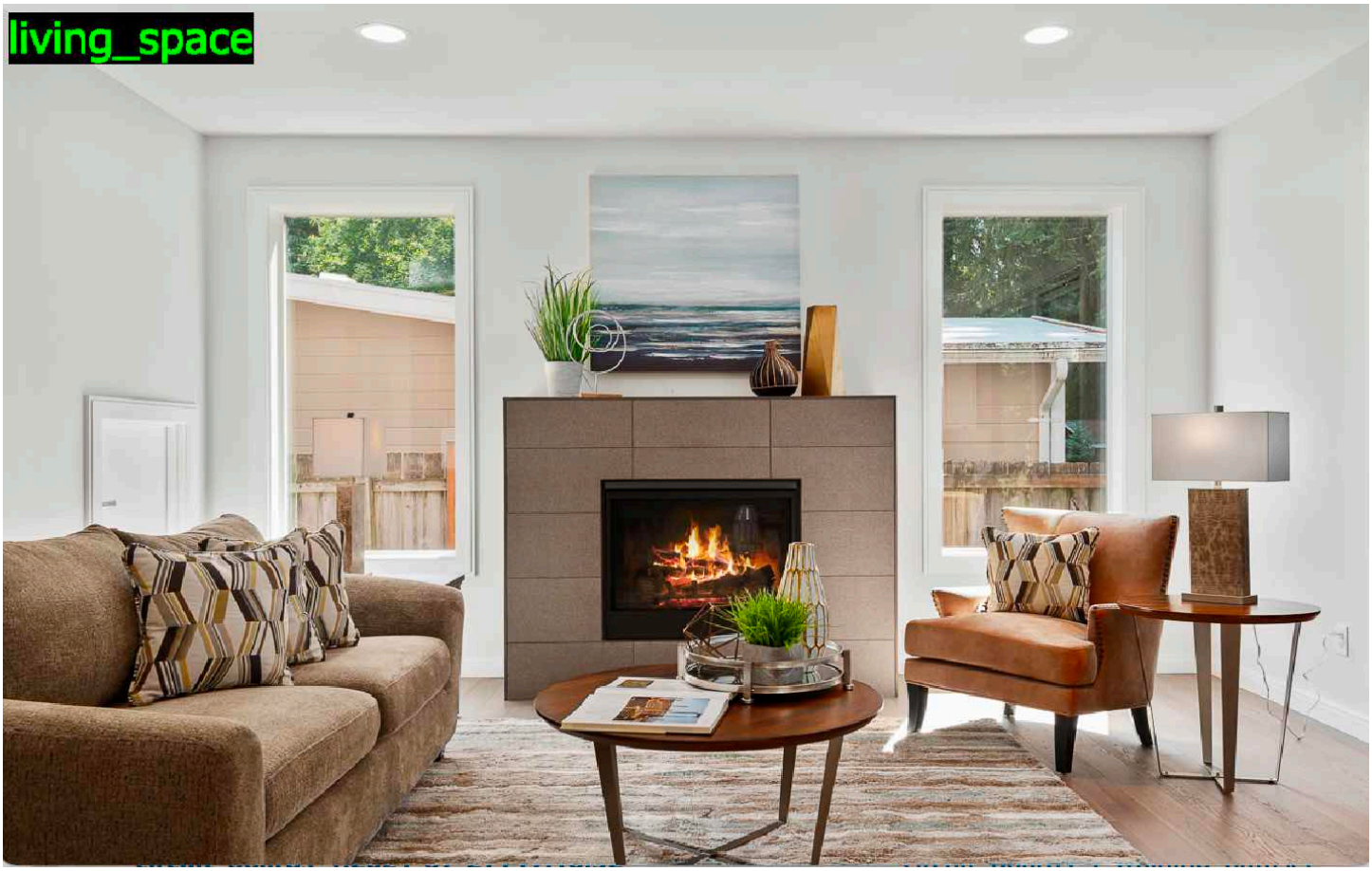
啟動模型

您開始您的模型呼叫StartProjectVersion。如需詳細資訊，請參閱啟動 Amazon Rekognition 自訂標籤模型 (SDK)。

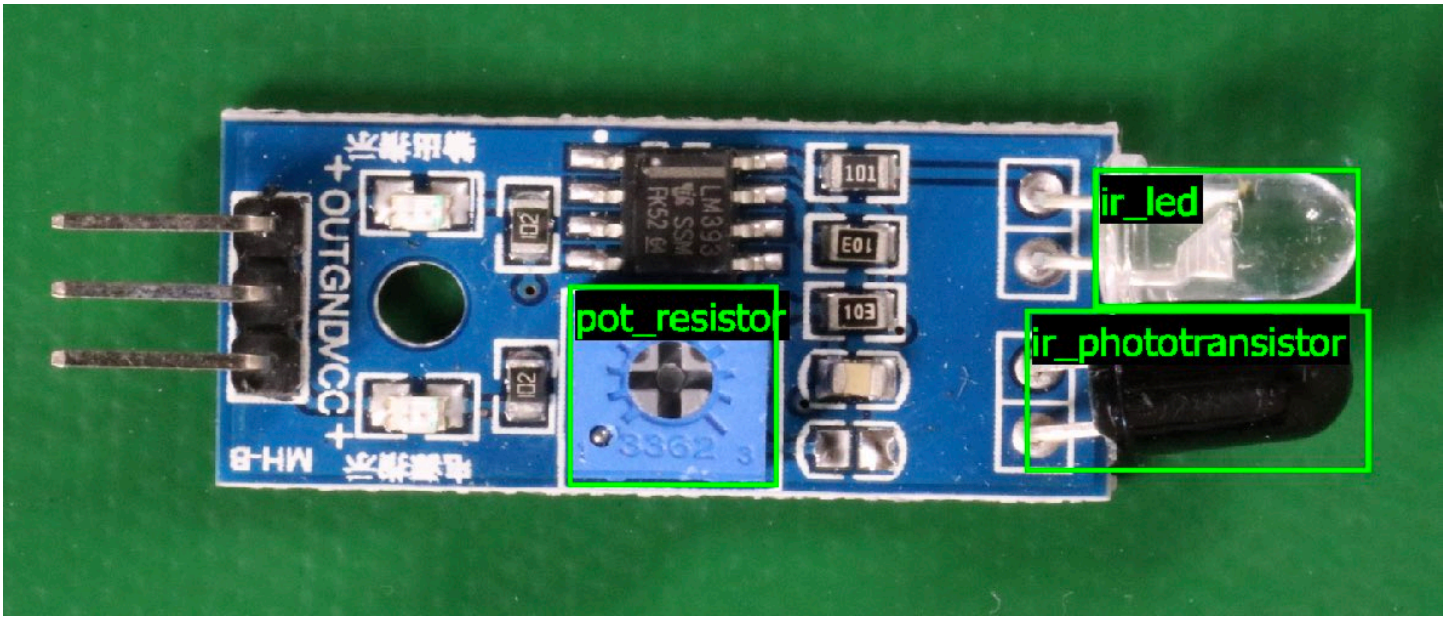
分析影像

若要使用模型分析影像，請使用DetectCustomLabels API。您可以指定本機映像，或存放在 S3 儲存貯體中的映像。此操作還需要您想要使用的模型的 Amazon Resource Name (ARN)。

如果您的模型找到物件、場景和概念，則回應會包含在影像中找到的影像層級標籤清單。例如，下圖展示使用「房間」範例專案中找到的影像層級標籤。



如果模型找到物件位置，則回應會包含影像中已標示的邊界方框清單。邊界方框代表物件在影像上的位置。您可以使用邊界方框資訊，在物件周圍繪製邊界方框。例如，以下影像展示了使用「電路板範例」專案找到的電路板零件周圍的邊界框。



如需詳細資訊，請參閱使用經過培訓的模型分析圖像。

停止模型

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 Amazon Rekognition 自訂標籤主控台或使用`StopProjectVersion` API 停止模型。如需詳細資訊，請參閱停止 Amazon Rekognition 自訂標籤模型。

停止您的模型（控制台）

若要使用主控台停止執行中的模型，請參閱停止 Amazon Rekognition 自訂標籤模型 (主控台)。

停止您的模型 (SDK)

若要停止執行中的模型，請呼叫StopProjectVersion。如需詳細資訊，請參閱 停止 Amazon Rekognition 自訂標籤模型 (SDK)。

開始使用亞馬遜自訂標籤

在開始這些之前開始使用說明，我們建議您閱讀了解 Amazon Rekognition 自訂標籤。

您可以使用 Amazon Rekognition 自訂標籤來訓練機器學習模型。訓練過的模型會分析影像，找出符合您業務需求的物件、場景和概念。例如，您可以訓練模型來分類房屋影像，或尋找印刷電路板上電子零件的位置。

為了協助您開始使用，Amazon Rekognition 自訂標籤包含教學影片和範例專案。

> **Note**
>
> 如需有關的資訊AWSAmazon 認知自訂標籤支援的區域和端點，請參閱重新認知端點和配額。

教學影片

這些影片向您展示如何使用 Amazon Rekognition 自訂標籤來訓練和使用模型。

檢視自學課程視訊的步驟

1. 登入AWS Management Console並在以下位置打開亞馬遜重新認知控制台https://console.aws.amazon.com/rekognition/。
2. 在左窗格中，選擇使用自訂標籤。顯示亞馬遜自訂標籤登陸頁面。如果您沒有看到使用自訂標籤，檢查是否AWS地區您正在使用支持亞馬遜自定義標籤。
3. 在導覽窗格中，選擇開始使用。
4. 在什麼是亞馬遜自定義標籤？中，選擇要觀看概覽視頻的視頻。
5. 在導覽窗格中，選擇教程。
6. 在「」教程頁面上，選擇您要觀看的教程視頻。

範例專案

Amazon 自訂標籤提供下列範例專案。

影像分類

影像分類專案 (房間) 會訓練在影像中尋找一或多個家庭位置的模型，例如後院,廚房，以及露台。訓練和測試影像代表單一位置。每個影像都標有單一影像層級標籤，例如廚房,露台，或生活空間。對於分析的影像，訓練過的模型會從用於訓練的影像層級標籤集傳回一個或多個相符的標籤。例如，模型可能會找到標籤生活空間在下圖中。如需詳細資訊，請參閱尋找物件、場景和概念。

多標籤圖像分類

多標籤圖像分類項目（Flowers）訓練一個模型，將花卉圖像分類為三個概念（花卉類型，葉子存在和生長階段）。

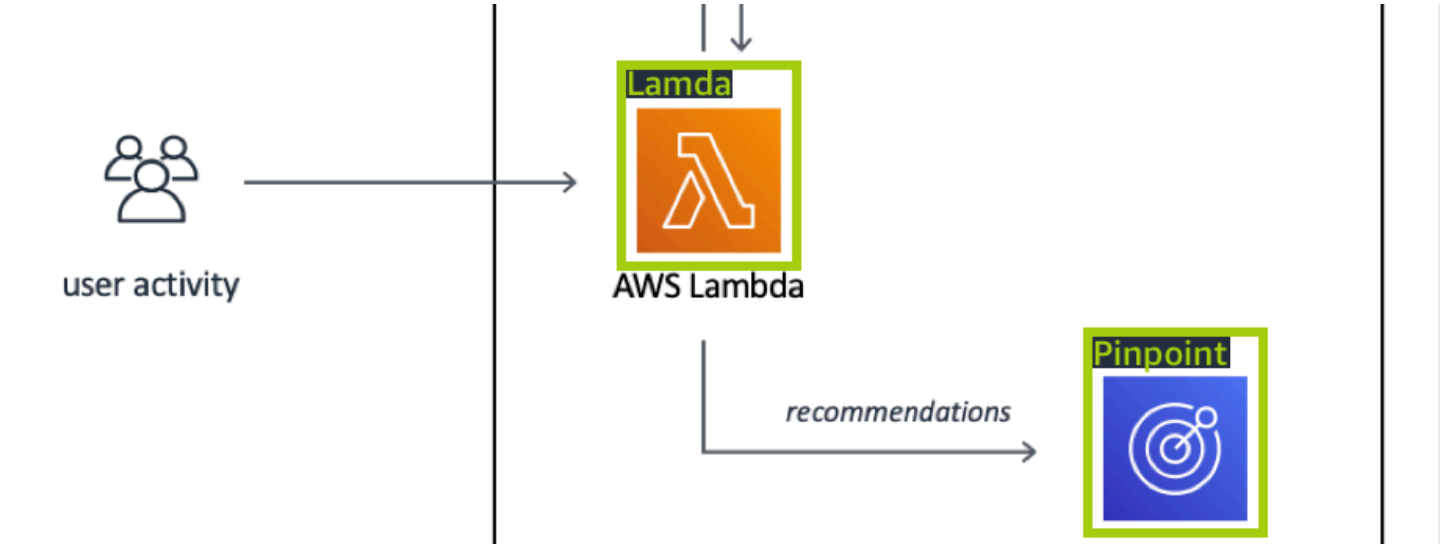
訓練和測試影像都有每個概念的影像層級標籤，例如山茶花對於花卉類型，與 _ 葉用於有葉子的花，和完全成長對於一朵完全生長的花。

對於已分析的影像，訓練過的模型會從用於訓練的影像層級標籤集傳回相符的標籤。例如，模型返回標籤地中海大戟和與 _ 葉對於下面的圖像。如需詳細資訊，請參閱尋找物件、場景和概念。

品牌檢測

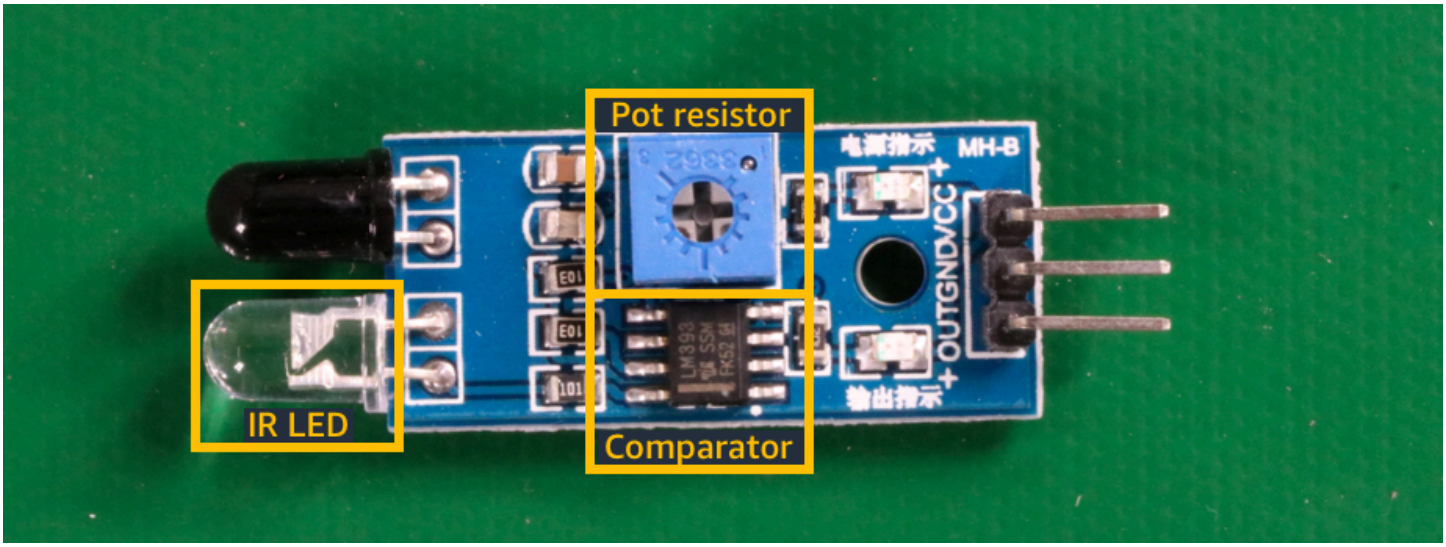
品牌檢測項目（徽標）訓練模型找到某些位置的模型AWS標誌，例如亚马逊短信，以及拉姆達。訓練圖像僅屬於徽標，並具有單個圖像級別標籤，例如lambda或者文字。也可以使用具有品牌位置邊框

的訓練圖像來訓練品牌檢測模型。測試影像標示了邊界方框，代表標誌在自然位置 (例如建築圖) 的位置。訓練過的模型會找到標誌，並針對找到的每個標誌傳回一個標記的邊界方框。如需詳細資訊，請參閱尋找品牌位置。



物件本地化

物件本地化專案 (電路板) 會訓練尋找印刷電路板上零件位置的模型，例如比較器或一個紅外線發光二極管。訓練和測試影像包括圍繞電路板零件的邊界框，以及用來識別邊界框內零件的標籤。標籤名稱是ir_ 光電晶體管,IR_ 領導,電阻器，以及比較器。訓練過的模型會尋找電路板零件，並針對找到的每個電路零件傳回一個標示的邊界。如需詳細資訊，請參閱尋找物件位置。



使用範例專案

這些入門指示說明如何使用 Amazon Rekognition 自訂標籤為您建立的範例專案來訓練模型。它還向您展示如何啟動模型並使用它來分析影像。

建立範例專案

若要開始使用，請決定要使用的專案。如需詳細資訊，請參閱步驟 1：選擇範例專案。

Amazon Rekognition 自訂標籤使用資料集來訓練和評估 (測試) 模型。資料集會管理影像和識別影像內容的標籤。範例專案包括訓練資料集，以及標示所有影像的測試資料集。在訓練模型之前，您不需要進行任何變更。範例專案顯示 Amazon Rekognition 自訂標籤使用標籤來訓練不同類型模型的兩種方式。

- 影像層級— 標籤可識別代表整個影像的物件、場景或概念。
- 邊界框— 標籤可識別邊界方框的內容。邊界方框是一組圍繞影像中物件的影像座標。

稍後，當您使用自己的映像建立專案時，您必須建立訓練和測試資料集，並標記您的影像。如需詳細資訊，請參閱決定您的型號類型。

訓練模型

在 Amazon Rekognition 自訂標籤建立範例專案之後，您可以訓練模型。如需詳細資訊，請參閱步驟 2：訓練您的模型。訓練結束後，您通常會評估模型的效能。範例資料集中的影像已建立高效能模型，您不需要在執行模型之前評估模型。如需詳細資訊，請參閱改善訓練有素的 Amazon Rekognition 自訂標籤模型。

使用模型

接下來，開始模型。如需詳細資訊，請參閱步驟 3：啟動模型。

開始執行模型後，您可以使用它來分析新影像。如需詳細資訊，請參閱步驟 4：使用模型分析圖像。

我們會根據模型執行的時間量向您收費。使用完範例模型後，您應該停止模型。如需詳細資訊，請參閱步驟 5：停止模型。

下一步驟

當你準備好了，你可以創建自己的項目。如需詳細資訊，請參閱步驟 6：後續步驟。

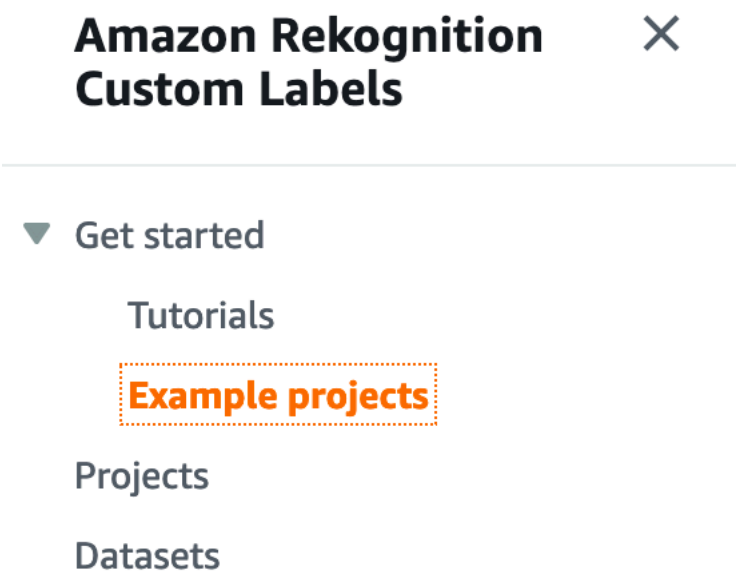
步驟 1：選擇範例專案

在此步驟中，您可以使用選擇範例專案。然後，Amazon Rekognition 自訂標籤會為您建立專案和資料集。專案會管理用來訓練模型的檔案。如需詳細資訊，請參閱管理 Amazon Rekognition 自訂標籤專案。資料集包含用於訓練和測試模型的影像、指定的標籤和邊界方框。如需詳細資訊，請參閱the section called “管理資料集”。

若要取得有關範例專案的資訊，請參閱範例專案。

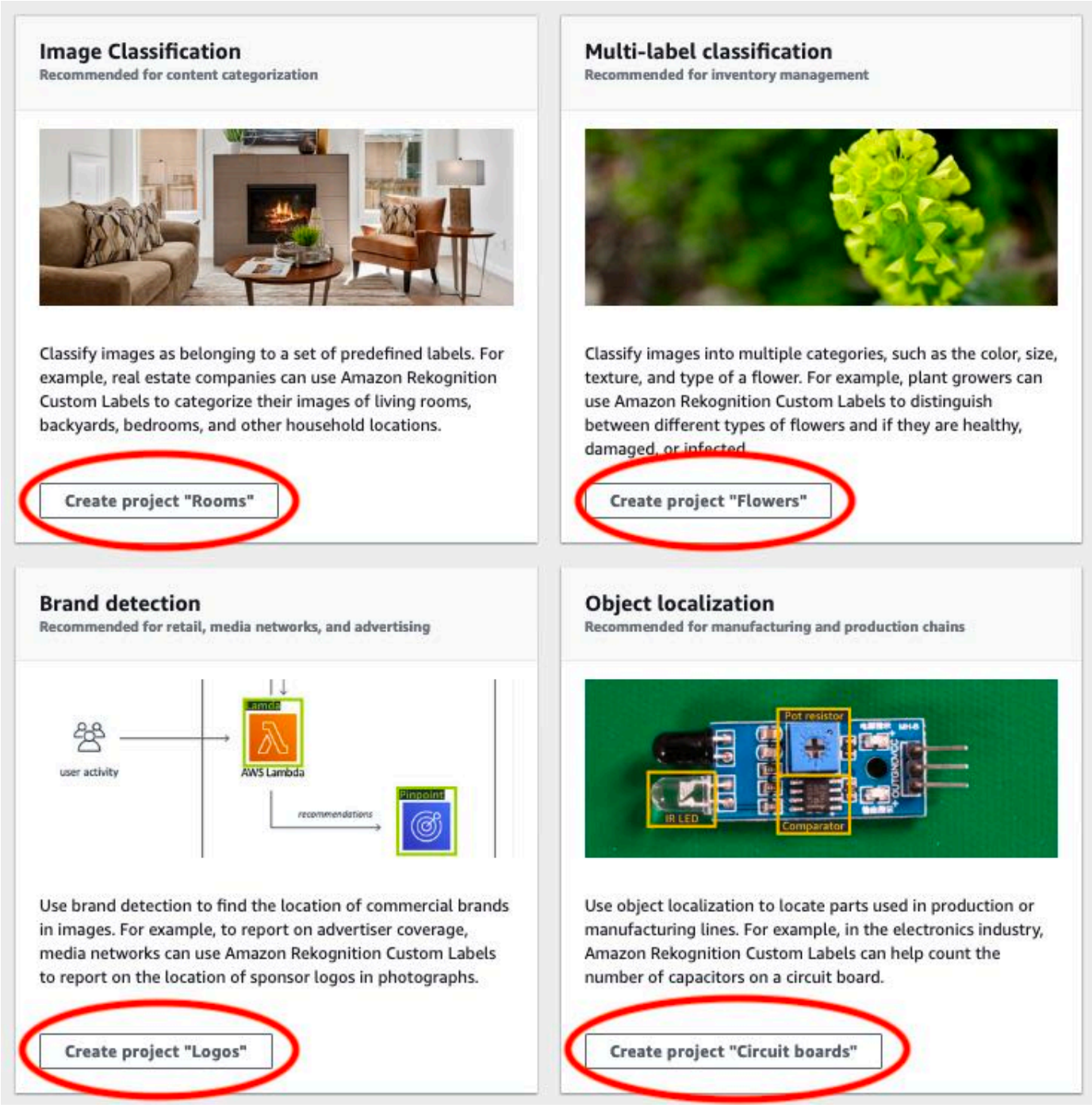
選擇範例專案

1. 登入AWS Management Console並在以下位置打開亞馬遜重新認知控制台https://console.aws.amazon.com/rekognition/。
2. 在左窗格中，選擇使用自訂標籤。顯示亞馬遜自訂標籤登陸頁面。如果您沒有看到使用自訂標籤，檢查是否AWS地區您正在使用支持亞馬遜自定義標籤。
3. 選擇 Get started (開始使用)。

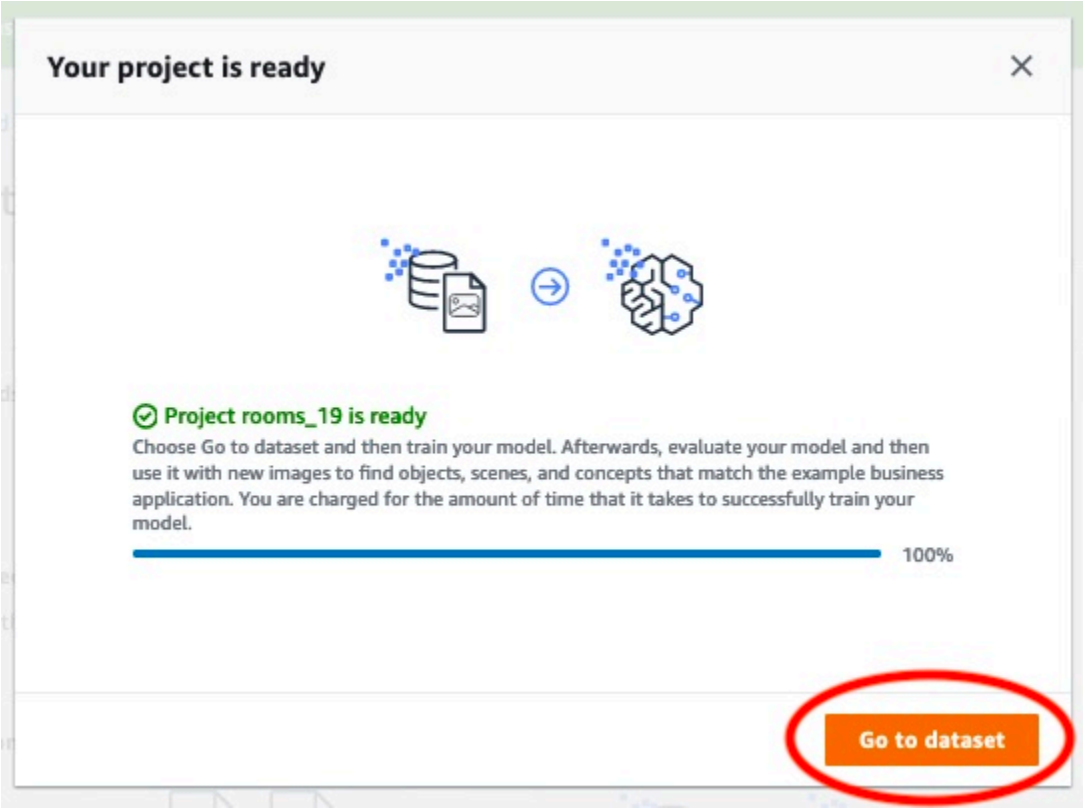


4. 在探索範例專案，選擇試用範例專案。
5. 決定您要使用的項目並選擇建立專案」####「在示例部分中。然後，Amazon 重新認知自訂標籤會為您建立範例專案。

> **Note**
>
> 如果這是您第一次以目前的方式開啟主控台AWS區域，首次設定顯示對話方塊。請執行下列動作：
>
> 1. 請注意顯示的亞馬遜 S3 存儲桶的名稱。
> 2. 選擇繼續讓亞馬遜自定義標籤代表您創建一個亞馬遜 S3 存儲桶（控制台存儲桶）。



6. 專案準備就緒後，請選擇前往資料集。

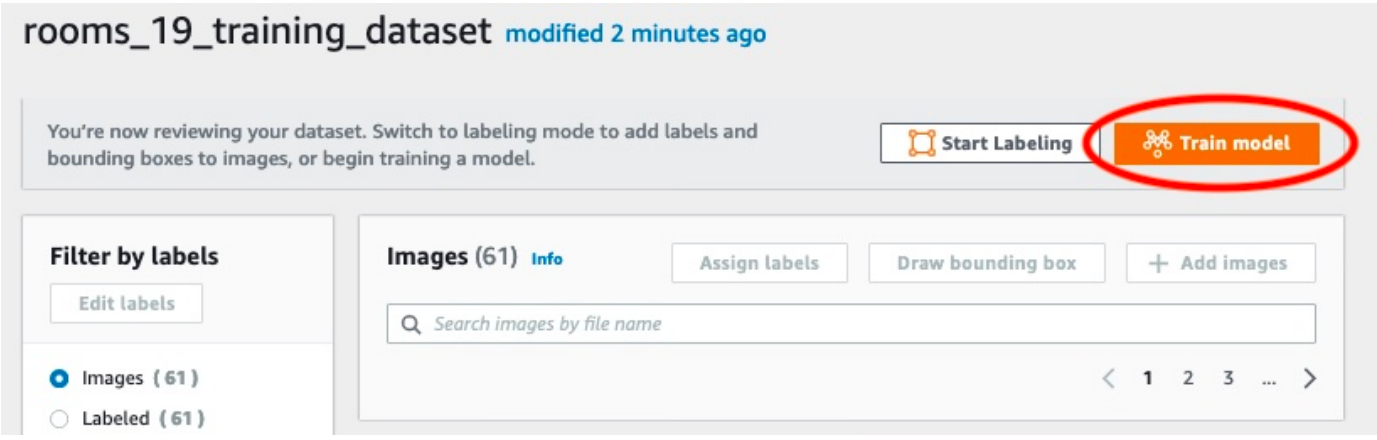


步驟 2：訓練您的模型

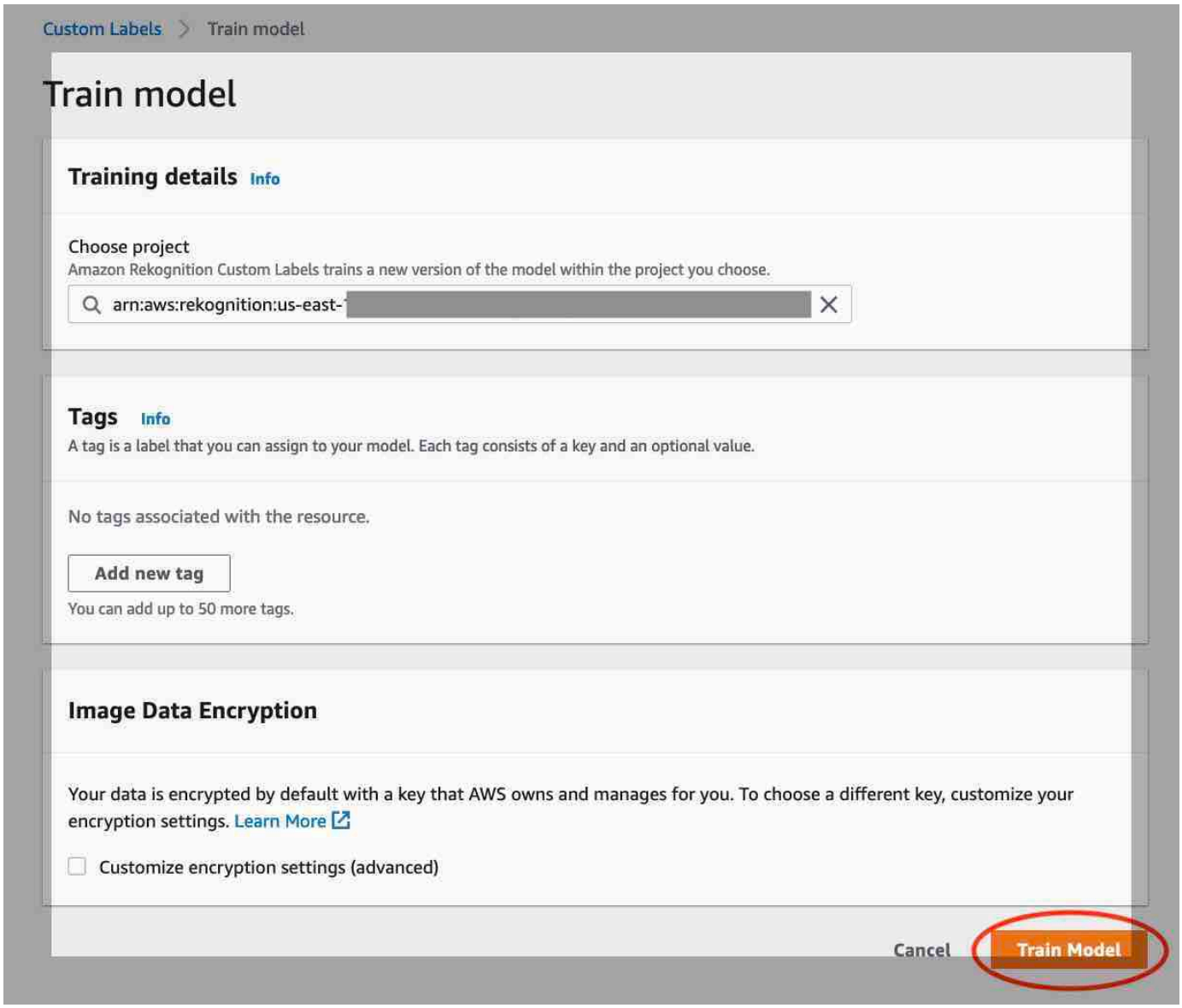
在此步驟中，您將訓練模型。訓練和測試資料集會自動為您設定。訓練成功完成後，您可以查看整體評估結果，以及個別測試影像的評估結果。如需詳細資訊，請參閱訓練 Amazon Rekognition 自訂標籤模型。

訓練您的 模型

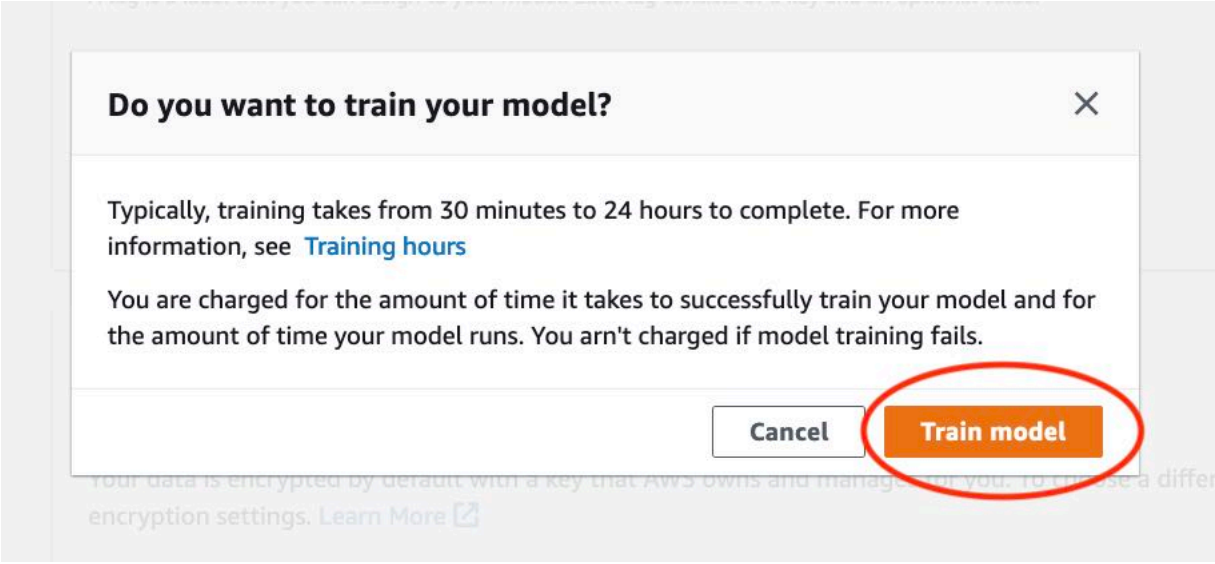
1. 在資料集頁面上，選擇火車模型。



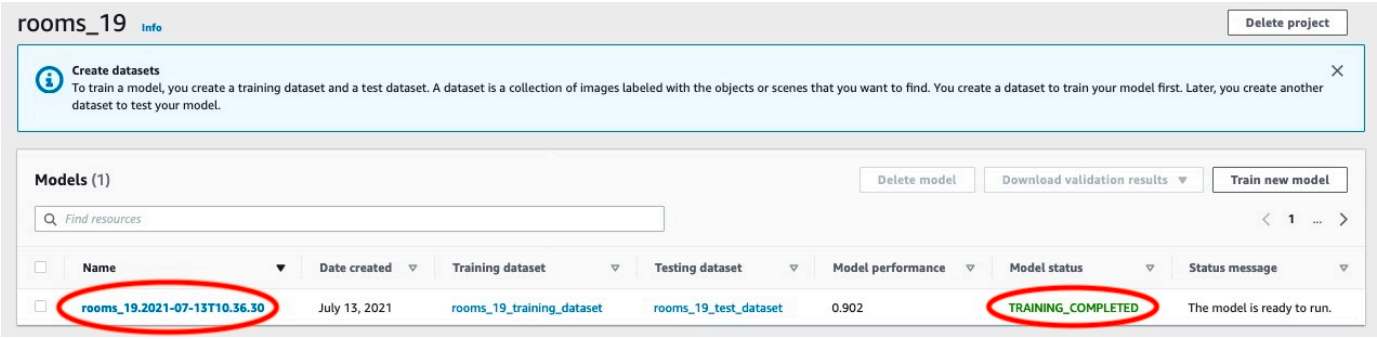
2. 在「」火車模型頁面，選擇火車模型。您的項目的亞馬遜資源名稱（ARN）位於選擇專案編輯方塊。



3. 在你想訓練你的模型嗎？」對話方塊中，選擇火車模型。

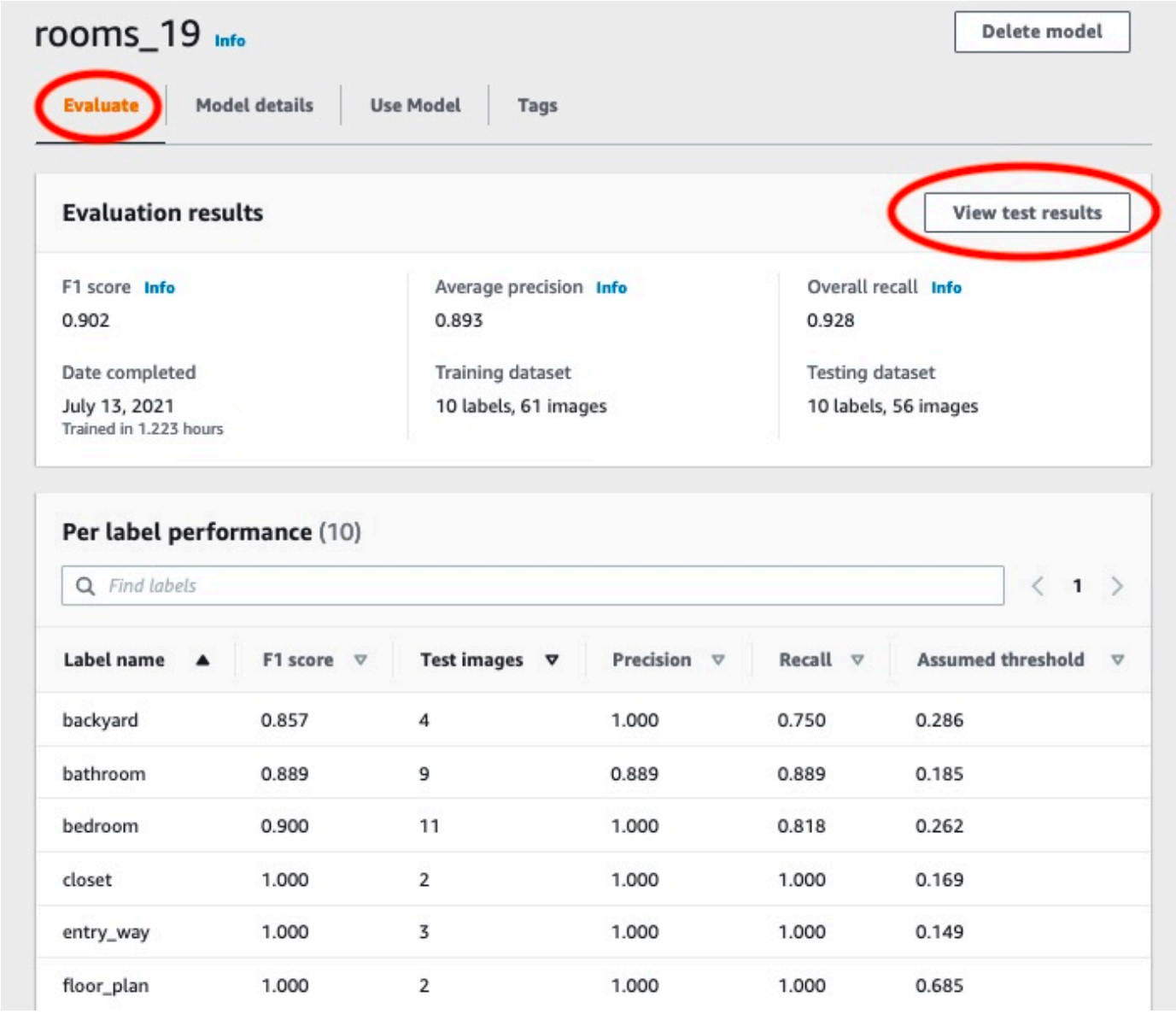


4. 訓練完成後，選擇模型名稱。模型狀態為時，訓練已完成訓練 (_已完成)。

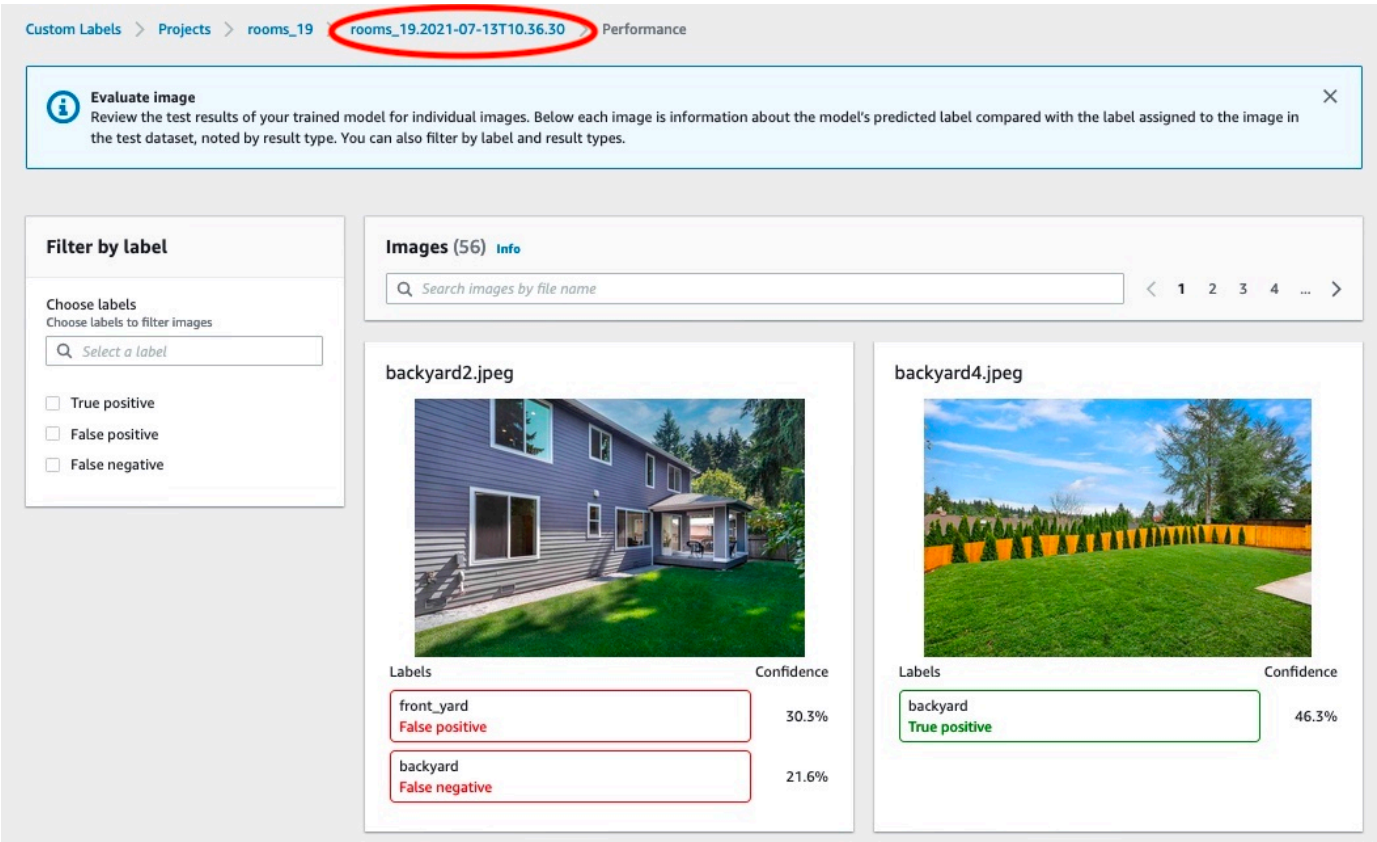


5. 選擇合適的評估按鈕以查看評估結果。如需有關評估模型的資訊，請參閱改善訓練有素的 Amazon Rekognition 自訂標籤模型。

6. 選擇檢視測試結果以查看個別測試影像的結果。



7. 檢視測試結果後，選擇要返回模型頁面的型號名稱。



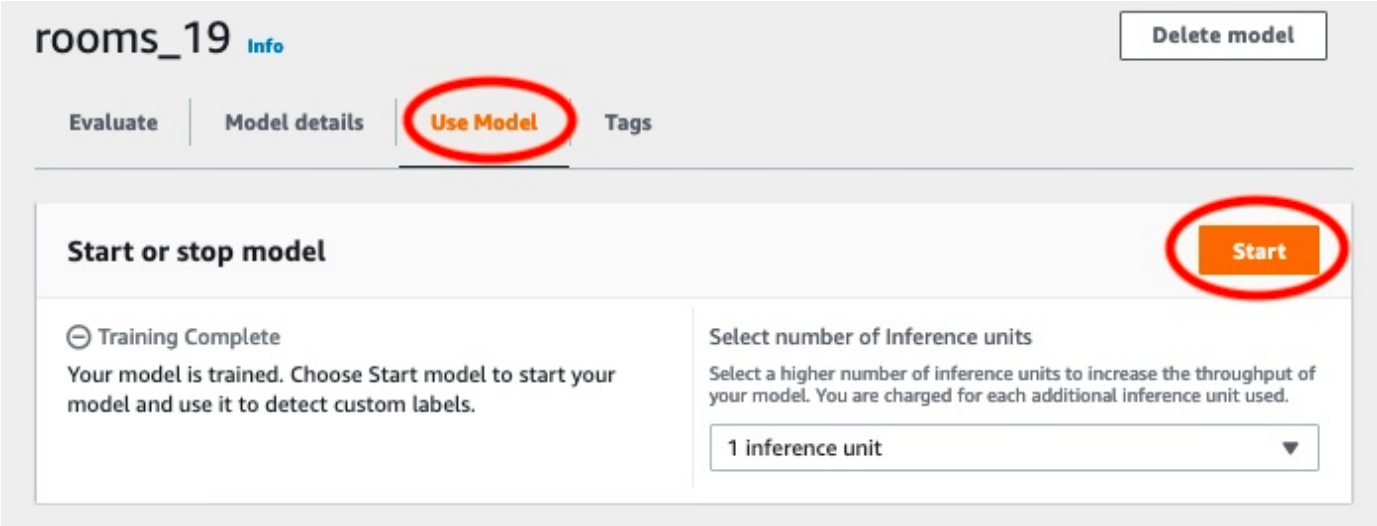
步驟 3：啟動模型

在此步驟中，您可以啟動模型。模型啟動後，您可以使用它來分析影像。

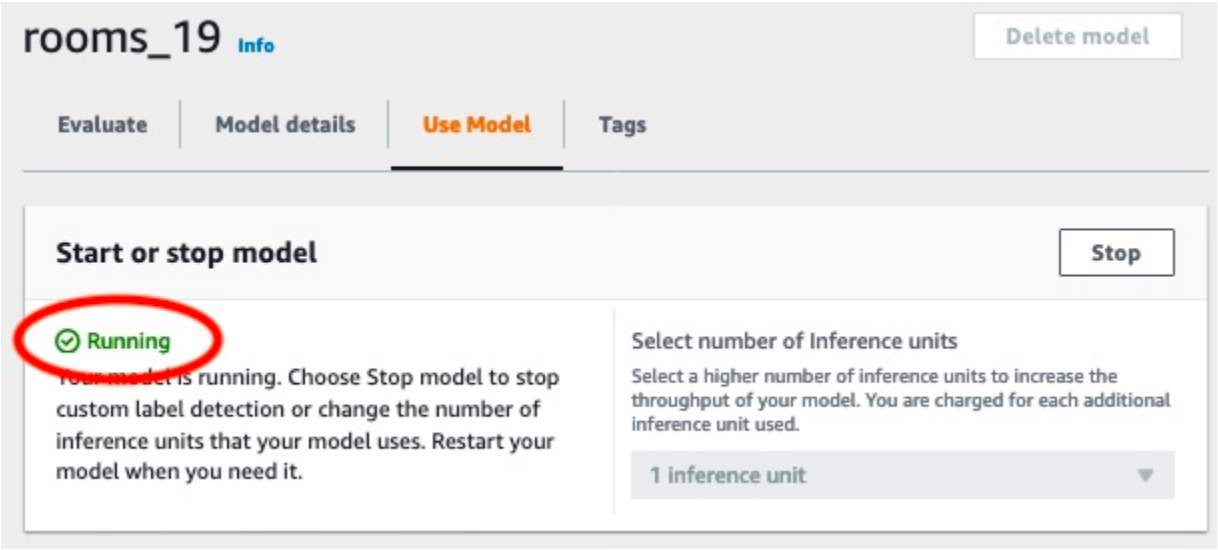
我們會根據模型執行的時間量向您收費。如果您不需要分析影像，請停止模型。您可以稍後重新啟動模型。如需詳細資訊，請參閱執行培訓過的 Amazon Rekognition 自訂標籤模型。

開始您的模型

1. 選擇合適的使用模型模型頁面上的標籤。
2. 在啟動或停止模型區段執行下列動作：
   a. 選擇 Start (啟動)。
   b. 在開始模型」對話方塊中，選擇開始。



3. 等到模型運行。模型正在運行時的狀態啟動或停止模型部分是跑步。



4. 使用您的模型來分類圖像。如需詳細資訊，請參閱步驟 4：使用模型分析圖像。

步驟 4：使用模型分析圖像

您可以通過調用分析圖像DetectCustomLabelsAPI。在此步驟中，您可以使用`detect-custom-labels` AWS Command Line Interface(AWS CLI) 指令來分析範例影像。你得到了AWS CLI來自亞馬遜自訂標籤主控台的命令。控制台配置AWS CLI指令以使用您的模型。您只需要提供存放在 Amazon S3 儲存貯體中的映像。本主題提供可用於每個範例專案的影像。

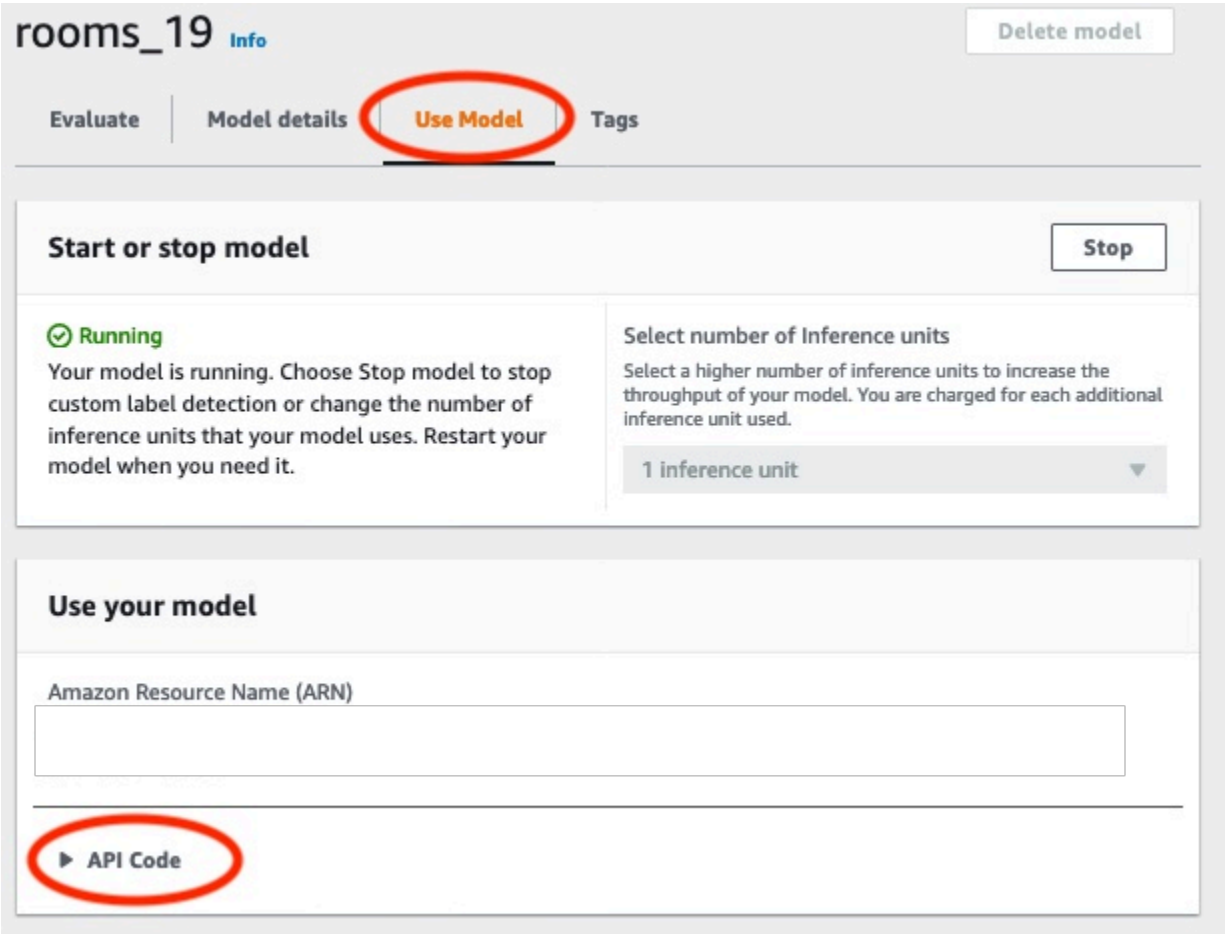
> **Note**
>
> 控制台還提供了 Python 示例代碼。

從輸出`detect-custom-labels`包括在影像中找到的標籤清單、邊界方框 (如果模型找到物件位置)，以及模型對預測準確度的可信度。

如需詳細資訊，請參閱使用經過培訓的模型分析圖像。

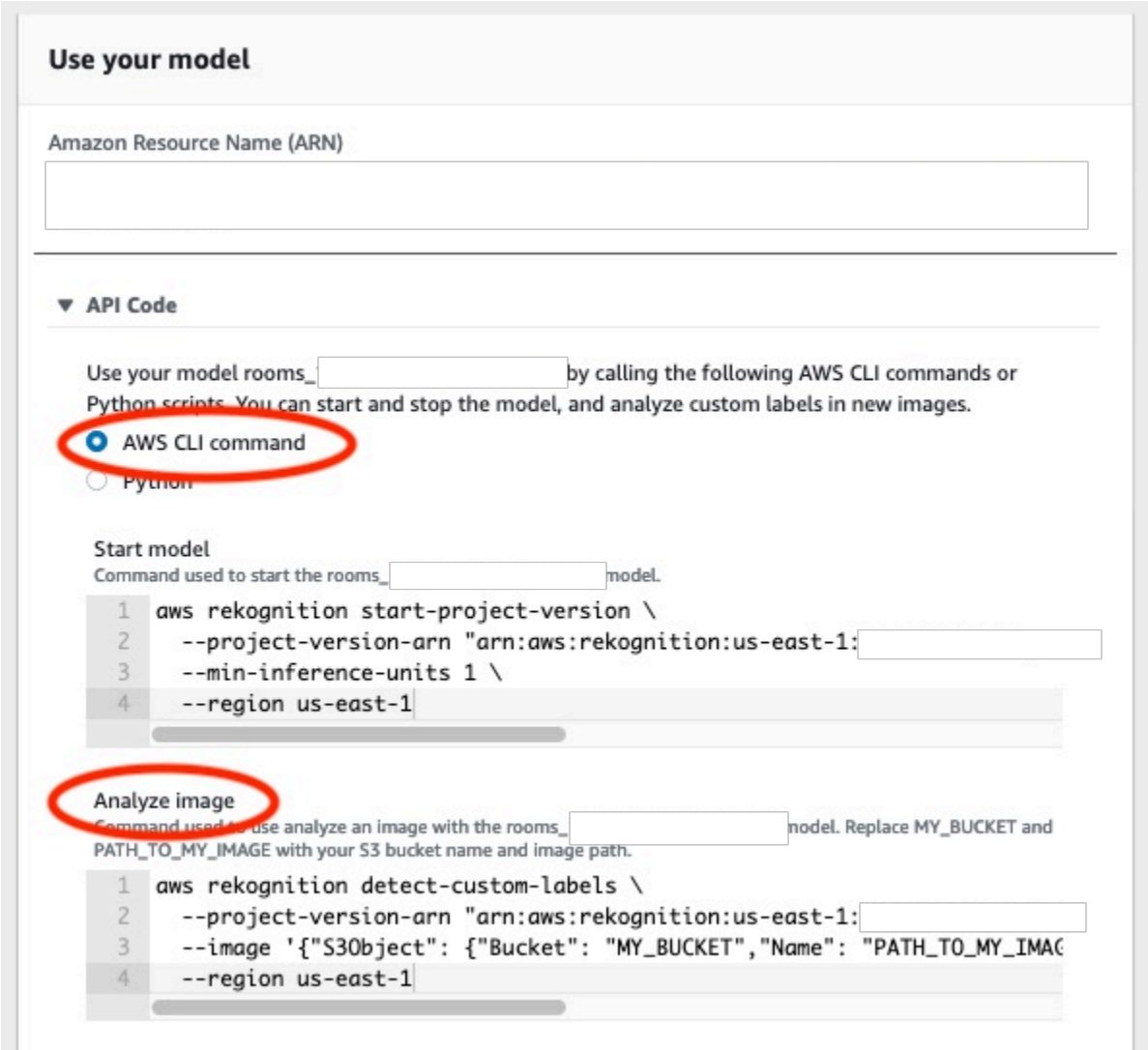
分析影像 (主控台) 的步驟

1. 如果您還沒有，請設定AWS CLI。如需相關指示，請參閱the section called “步驟 4：設定 AWS CLI 和 AWS SDKs”。
2. 如果您還沒有，請開始運行您的模型。如需詳細資訊，請參閱步驟 3：啟動模型。
3. 選擇合適的使用模型標籤，然後選擇API 程式碼。



4. 選擇命令。

5. 在分析影像區段中，複製AWS CLI呼叫的命令`detect-custom-labels`。



6. 將範例影像上傳到 Amazon S3 儲存貯體。如需相關指示，請參閱獲取示例圖像。

7. 於指令提示下，輸入AWS CLI您在上一個步驟中複製的指令。它應該看起來像下面的例子。

   的價值`--project-version-arn`應該是您模型的亞馬遜資源名稱（ARN）。的價值`--region`應該是AWS您在其中建立模型的區域。

   變更`MY_BUCKET`和`PATH_TO_MY_IMAGE`到您在上一個步驟中使用的 Amazon S3 儲存貯體和映像檔。

   如果您正在使用custom-labels-access配置文件獲取憑據，添加`--profile custom-labels-access`參數。

```
aws rekognition detect-custom-labels \
```

```
  --project-version-arn "model_arn" \
  --image '{"S3Object": {"Bucket": "MY_BUCKET","Name": "PATH_TO_MY_IMAGE"}}' \
  --region us-east-1 \
  --profile custom-labels-access
```

如果模型找到物件、場景和概念，則會從AWS CLI命令看起來應該類似於以下內容。Name是模型找到的影像層級標籤名稱。Confidence(0-100) 是模型對預測準確性的信心。

```
{
    "CustomLabels": [
        {
            "Name": "living_space",
            "Confidence": 83.41299819946289
        }
    ]
}
```

如果模型找到物件位置或找到品牌，則會傳回標示的邊界方框。BoundingBox包含物件周圍方塊的位置。Name是模型在邊界方框中找到的物件。Confidence是模型對邊界方框包含物件的信賴度。

```
{
    "CustomLabels": [
        {
            "Name": "textract",
            "Confidence": 87.7729721069336,
            "Geometry": {
                "BoundingBox": {
                    "Width": 0.198987677693367,
                    "Height": 0.31296101212501526,
                    "Left": 0.07924537360668182,
                    "Top": 0.4037395715713501
                }
            }
        }
    ]
}
```

8. 繼續使用模型分析其他影像。如果您不再使用模型，請停止該模型。如需詳細資訊，請參閱步驟 5：停止模型。

獲取示例圖像

您可以將下列影像與DetectCustomLabels操作。每個項目都有一個圖像。若要使用映像檔，請將它們上傳到 S3 儲存貯體。

若要使用範例影像

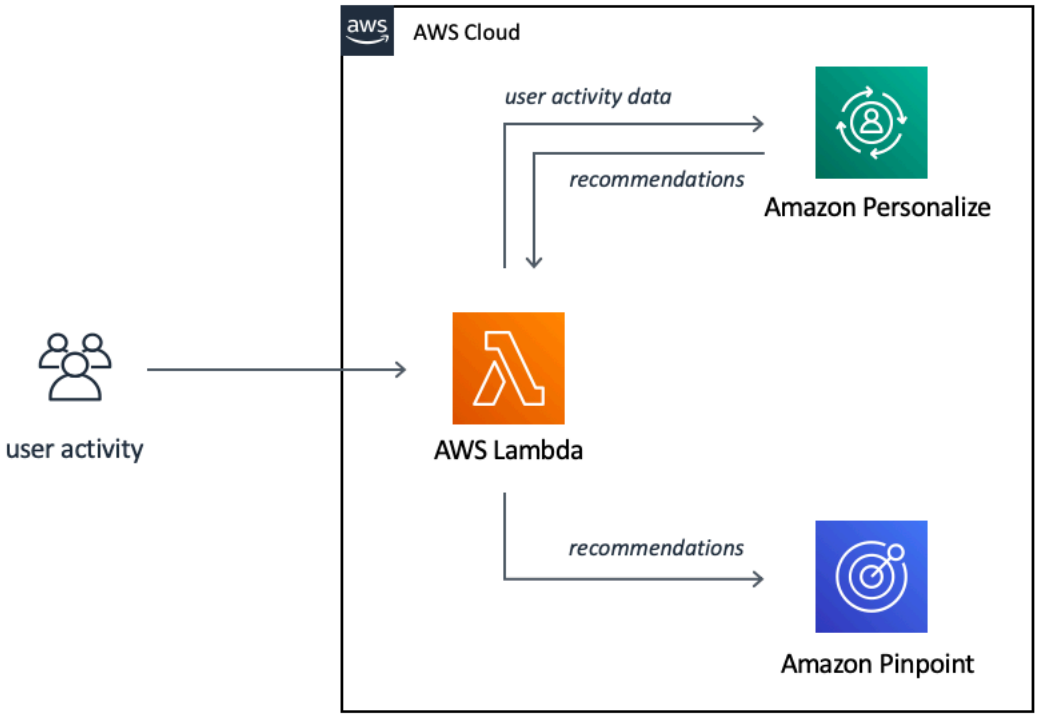
1. 在下列與您使用的範例專案相符的影像上按一下右鍵。然後選擇儲存影像將圖像保存到計算機。功能表選項可能會有所不同，具體取決於您使用的瀏覽器。
2. 將映像上傳到您擁有的 Amazon S3 儲存貯體AWS帳戶和在相同AWS您正在使用亞馬遜自訂標籤的區域。

   如需指示，請參閱將物件上傳到亞馬遜 S3在亞馬遜簡單存儲服務用戶指南。

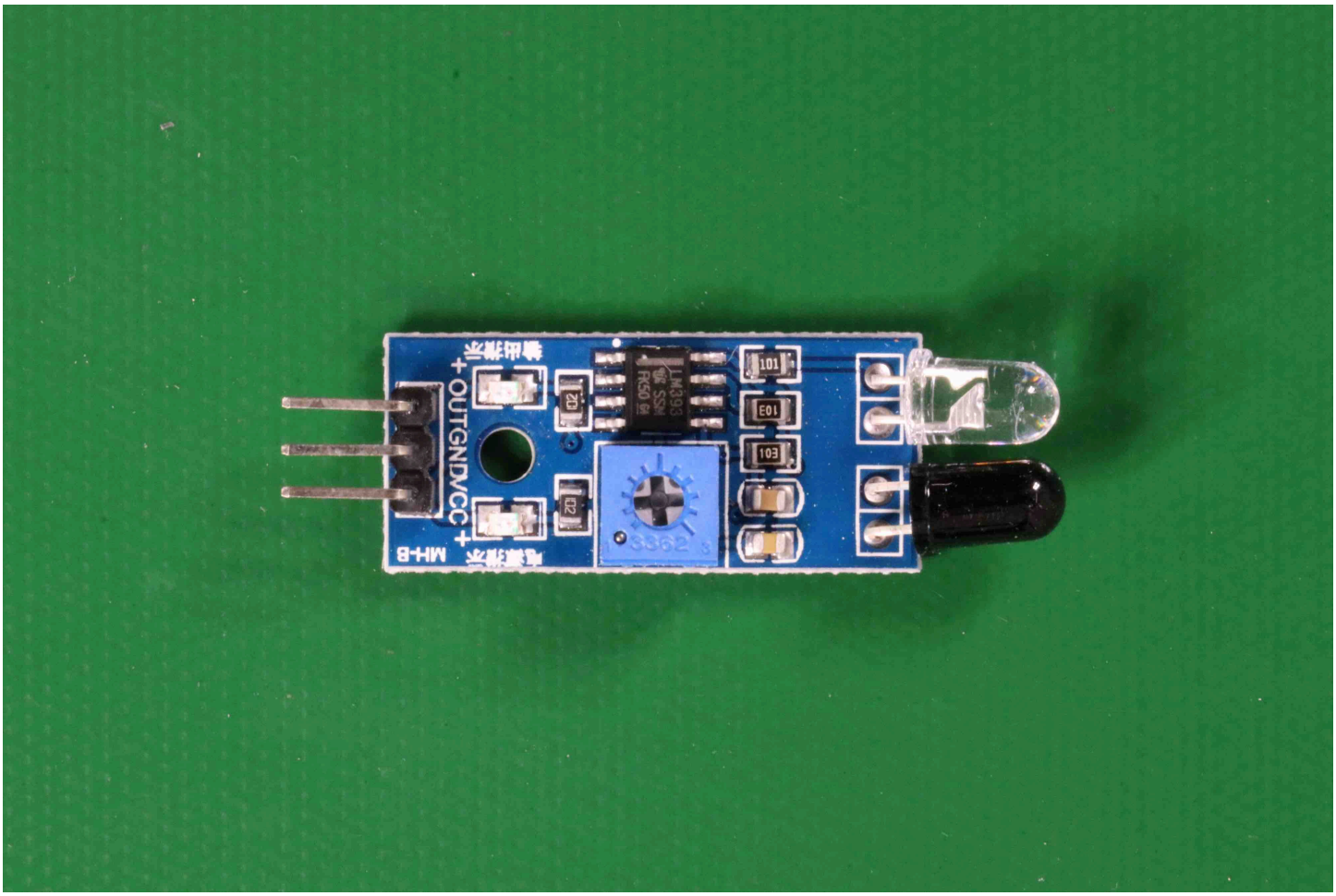
影像分類

多標籤分類

品牌檢測


aws
AWS Cloud
user activity data
recommendations
Amazon Personalize
user activity
AWS Lambda
recommendations
Amazon Pinpoint

物件本地化

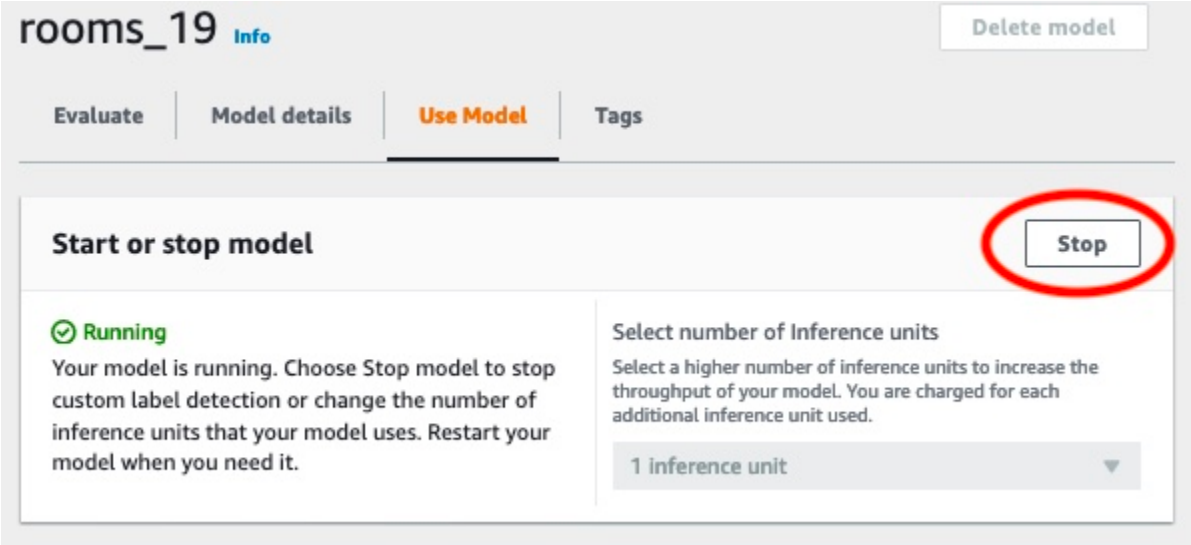


步驟 5：停止模型

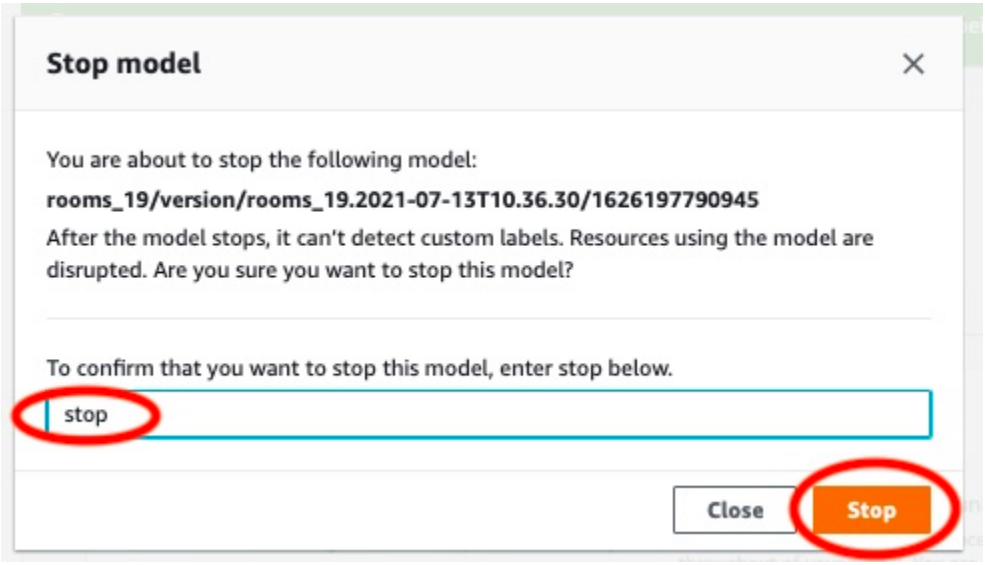
在此步驟中，您將停止執行模型。您需支付模型執行時間的費用。如果您已使用完模型，則應停止模型。

若要停止模型

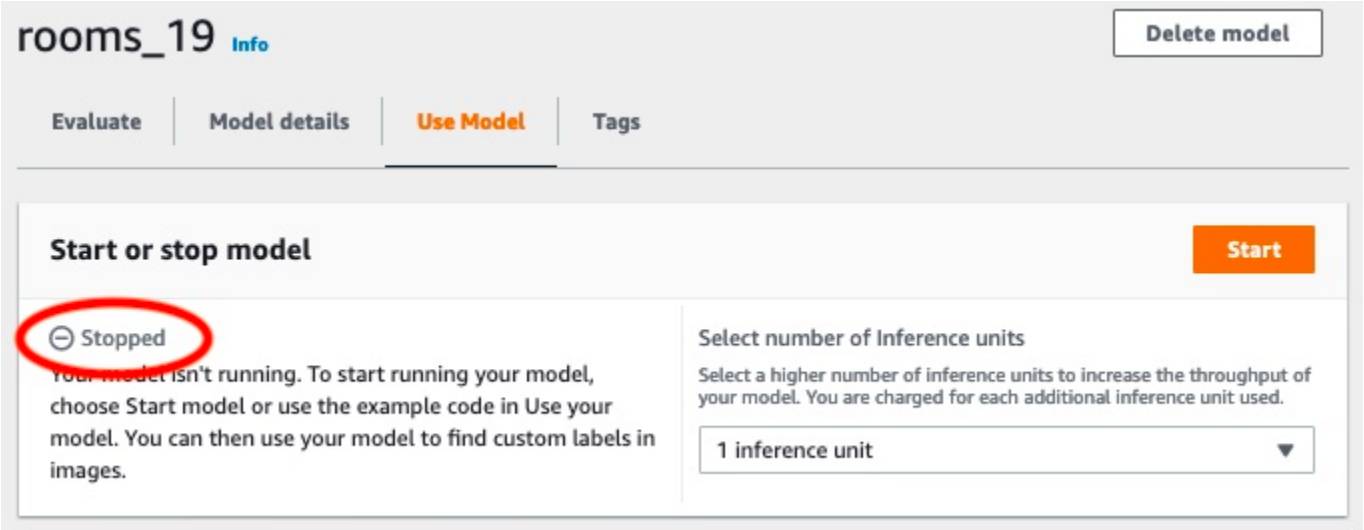
1. 在啟動或停止模型部分選擇停止。



2. 在停止模型」對話方塊中，輸入停以確認您要停止模型。



3. 選擇停止停止您的模型。模型已停止時的狀態啟動或停止模型部分是已停止。



步驟 6：後續步驟

完成範例專案的試用後，您可以使用自己的影像和資料集來建立自己的模型。如需詳細資訊，請參閱了解 Amazon Rekognition 自訂標籤。

使用下表中的標示資訊來訓練與範例專案類似的模型。

範例	訓練圖片	測試影像
圖像分類（客房）	每個影像 1 個影像層級標籤	每個影像 1 個影像層級標籤
多標籤分類 (花朵)	每張影像有多個影像層級標籤	每張影像有多個影像層級標籤
品牌檢測（徽標）	影像層級標籤 (您也可以使用標示的邊界方框)	標示的邊界方框
圖像定位（電路板）	標示的邊界方框	標示的邊界方框

該教學課程：分類影像示範如何為影像分類模型建立專案、資料集和模型。

如需建立資料集和訓練模型的詳細資訊，請參閱建立 Amazon Rekognition 型。

教學課程：分類影像

本教學課程說明如何為模型建立專案和資料集，以分類影像中的物件、場景和概念。模型會對整個影像進行分類。例如，透過遵循此自學課程，您可以訓練模型以辨識家庭位置，例如客廳或廚房。本自學課程還向您展示如何使用模型來分析影像。

在開始本教學課程之前，我們建議您先閱讀了解 Amazon Rekognition 自訂標籤。

在本教學課程中，您會從本機電腦上傳影像來建立訓練和測試資料集。稍後，您可以為訓練和測試資料集中的影像指派影像層級標籤。

您建立的模型會將影像分類為屬於您指派給訓練資料集影像的影像層級標籤集。例如，如果訓練資料集中的影像層級標籤集為`kitchen`,`living_room`,`patio`，以及`backyard`，模型可能會在單個圖像中找到所有這些圖像級標籤。

> **Note**
>
> 您可以為不同的目的建立模型，例如尋找影像上物件的位置。如需詳細資訊，請參閱決定您的型號類型。

步驟 1：收集您的圖像

您需要兩組圖像。一組可新增至訓練資料集。要新增至測試資料集的另一組。影像應代表您希望模型分類的物件、場景和概念。影像必須是 PNG 或 JPEG 格式。如需詳細資訊，請參閱準備影像。

您的訓練資料集應該至少有 10 個映像檔，測試資料集應該至少有 10 個映像檔。

如果您還沒有圖像，請使用圖像客房範例分類專案。建立專案後，訓練和測試映像會位於下列 Amazon S3 儲存貯體位置：

- 訓練圖片 —`s3://custom-labels-console-`*`region`*`-`*`numbers`*`/assets/rooms_`*`version number`*`_test_dataset/`
- 測試圖片 —`s3://custom-labels-console-`*`region`*`-`*`numbers`*`/assets/rooms_`*`version number`*`_test_dataset/`

`region`就是AWS您正在使用 Amazon 自訂標籤主控台的區域。`numbers`是控制台分配給存儲桶名稱的值。`Version number`是範例專案的版本號碼，從 1 開始。

下列程序會將 Rooms 專案中的影像儲存到名為的電腦上的本端資料夾`training`和`test`。

下載房間範例專案影像檔的步驟

1. 建立「房間」專案。如需詳細資訊，請參閱步驟 1：選擇範例專案。
2. 開啟命令提示字元，然後輸入以下指令以下載訓練影像。

```
aws s3 cp s3://custom-labels-console-region-numbers/assets/rooms_version
 number_training_dataset/ training --recursive
```

3. 在推薦提示下，輸入以下指令以下載測試影像。

```
aws s3 cp s3://custom-labels-console-region-numbers/assets/rooms_version
 number_test_dataset/ test --recursive
```

4. 將兩個影像從訓練資料夾移至您選擇的個別資料夾。您將使用圖像來嘗試訓練過的模型步驟 9：使用模型分析圖像。

步驟 2：決定您的課程

列出您希望模型查找的類。例如，如果您正在訓練模型以辨識房屋中的房間，則可以將下列影像分類為`living_room`。

每個類映射到一個圖像級標籤。稍後，您可以為訓練和測試資料集中的影像指派影像層級標籤。

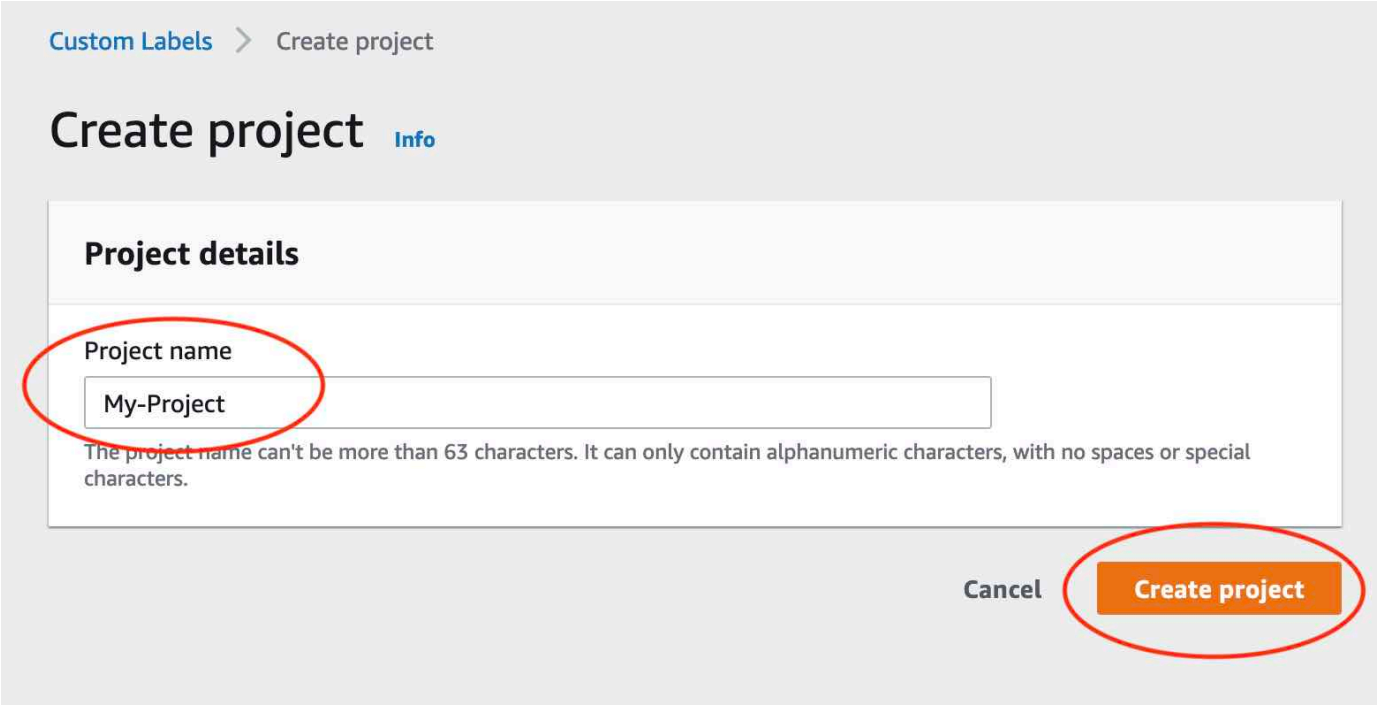
如果您使用的是 Rooms 範例專案中的影像，則影像層級的標籤為後院,浴室,臥室,壁櫥,入口路,樓板平面圖,前院,廚房,生活空間，以及露台。

步驟 3：建立專案

若要管理資料集和模型，請建立專案。每個專案都應處理單一使用案例，例如識別房屋中的房間。

若要建立專案 (主控台)

1. 如果您還沒有，請設定 Amazon Rekognition 自訂標籤主控台。如需詳細資訊，請參閱設定 Amazon Rekognition 自訂標籤。
2. 登入AWS Management Console並在以下位置打開亞馬遜重新認知控制台https://console.aws.amazon.com/rekognition/。
3. 在左窗格中，選擇使用自訂標籤。顯示亞馬遜自訂標籤登陸頁面。
4. 亞馬遜自定義標籤登陸頁面，選擇開始使用
5. 在左側導覽窗格中，選擇項目。
6. 在「項目」頁面上，選擇建立專案。
7. 在 Project name (專案名稱) 中，輸入您的專案名稱。
8. 選擇建立專案創建您的項目。



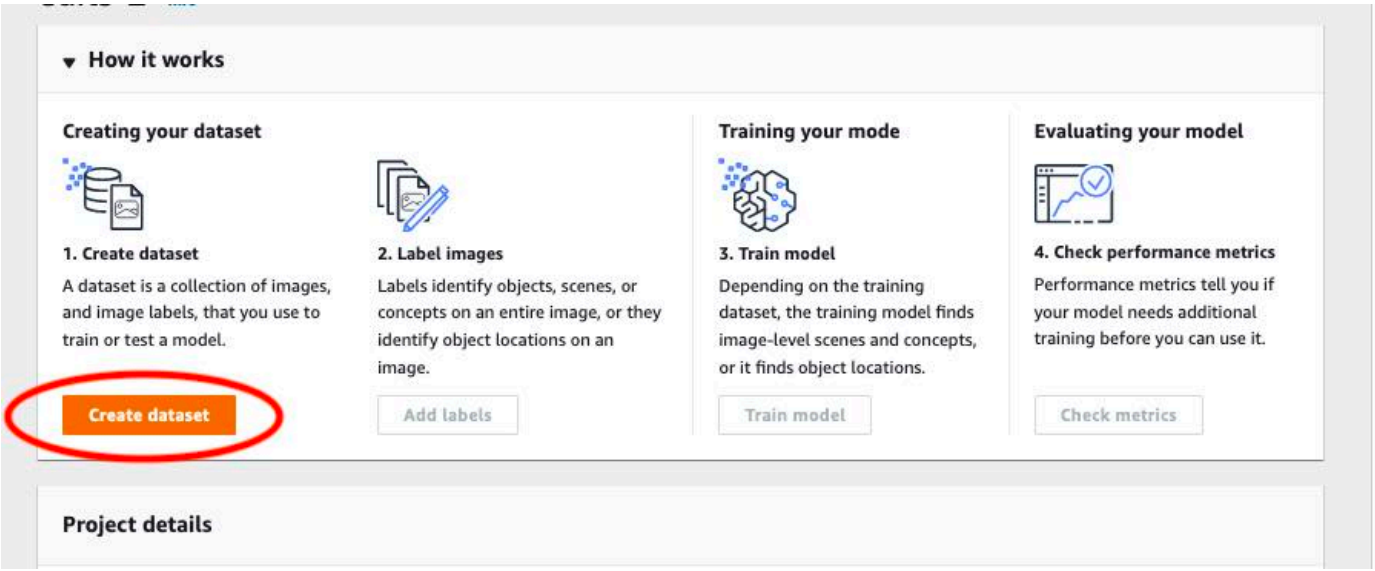
步驟 4：建立訓練和測試資料集

在此步驟中，您可以從本機電腦上傳影像，以建立訓練資料集和測試資料集。您一次最多可以上傳 30 張圖片。如果您有很多要上傳的影像，請考慮從 Amazon S3 儲存貯體匯入映像以建立資料集。如需詳細資訊，請參閱Amazon S3 儲存貯體。

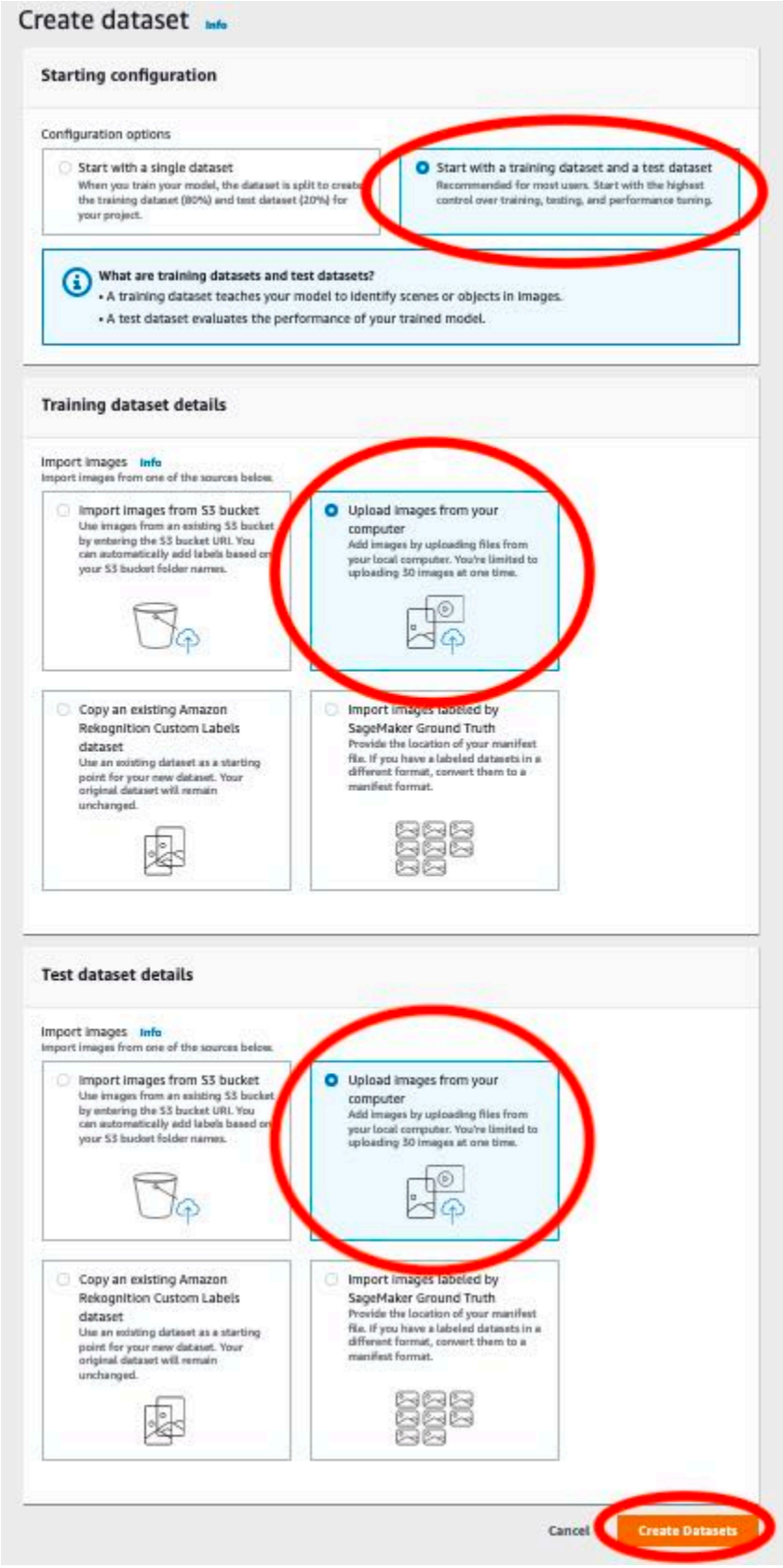
如需資料集的詳細資訊，請參閱管理資料集。

使用本機電腦上的影像建立資料集 (主控台)

1. 在專案詳細資訊頁面上，選擇建立資料集。

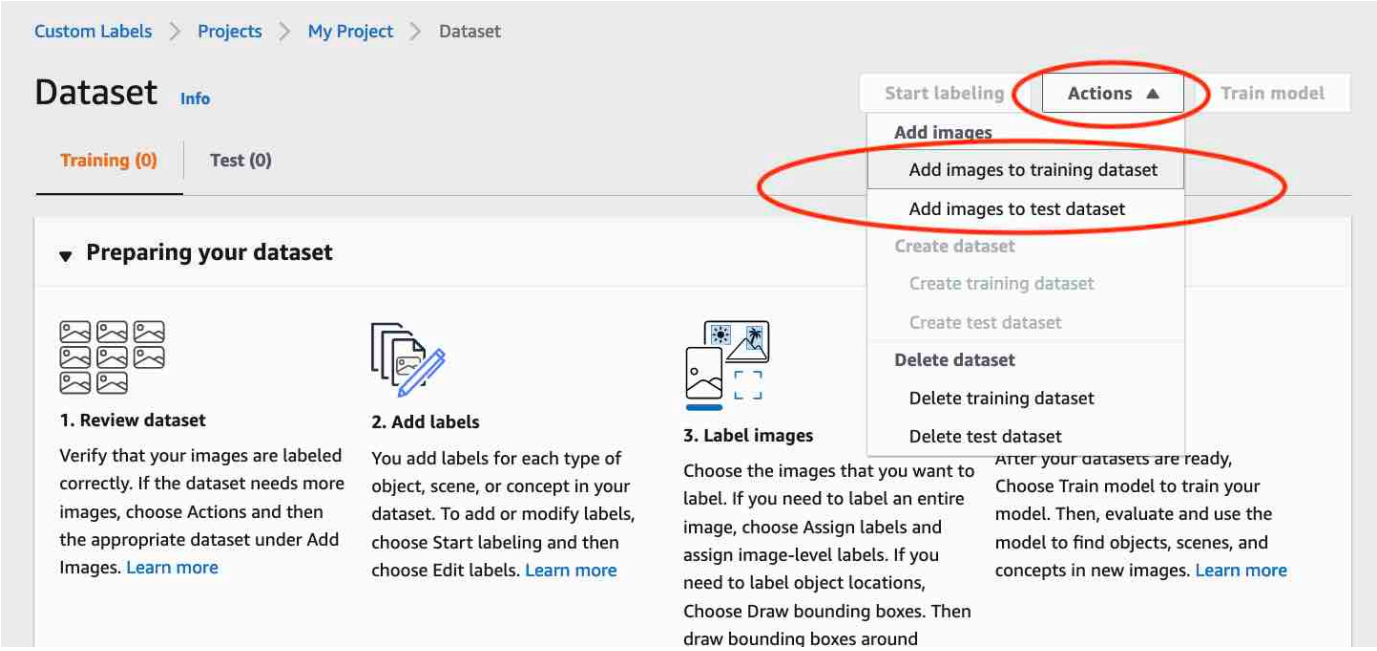


2. 在起始組態區段中，選擇從訓練資料集和測試資料集開始。
3. 在訓練資料集詳情區段中，選擇從您的電腦上傳圖片。
4. 在測試資料集詳情區段中，選擇從您的電腦上傳圖片。
5. 選擇建立資料集。

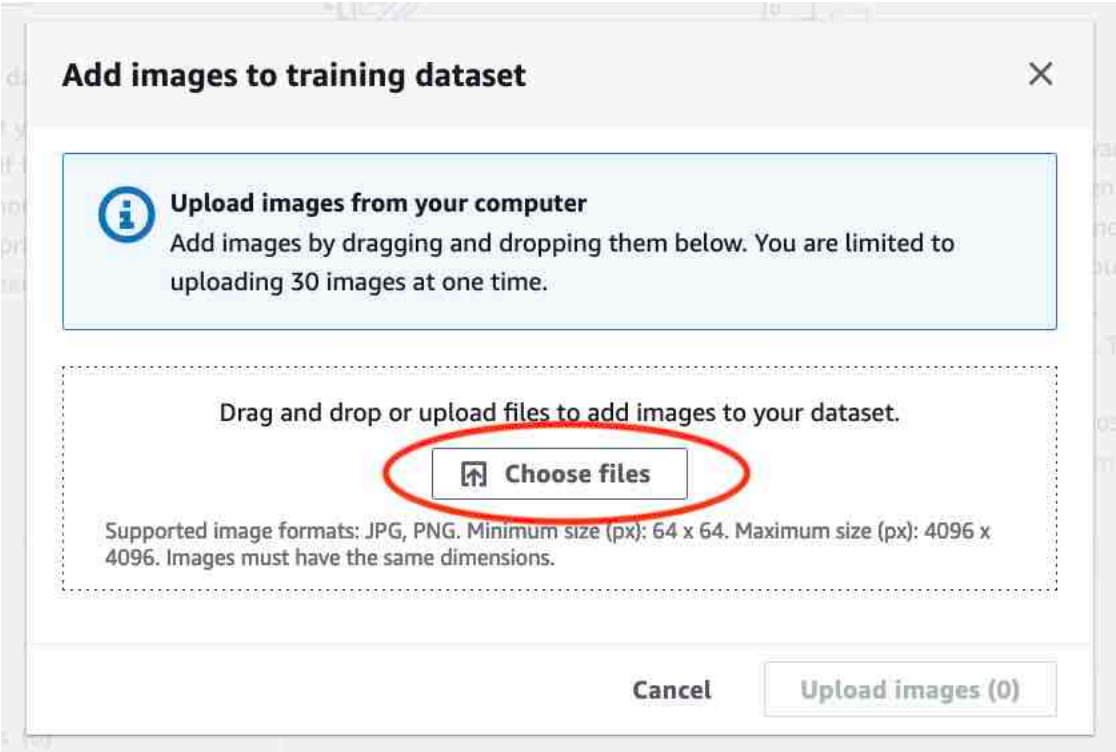


6. 會顯示資料集頁面，其中包含訓練標籤和測試相應資料集的索引標籤。

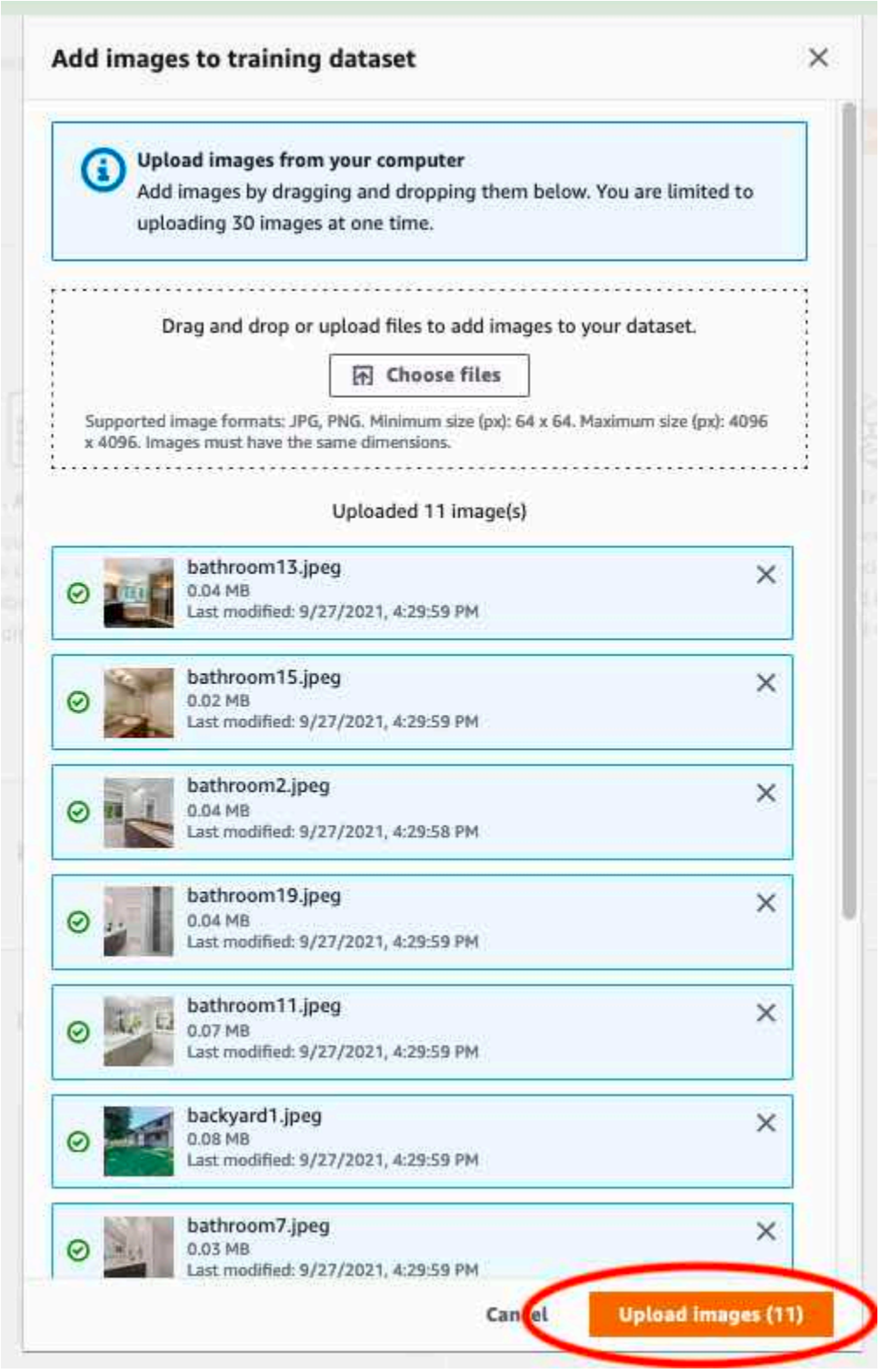
7. 在資料集頁面上，選擇訓練標籤。
8. 選擇動作然後選擇將影像新增至訓練資料集。



9. 在將影像新增至訓練資料集」對話方塊中，選擇選擇檔案。



10. 選擇您要上傳至資料集的影像。您一次最多可以上傳 30 張圖片。
11. 選擇上傳圖片。Amazon Rekognition 自訂標籤可能需要幾秒鐘的時間，才能將映像檔新增至資料集。



12. 如果您有更多影像要新增至訓練資料集，請重複步驟 9-12。
13. 選擇 測試 標籤。

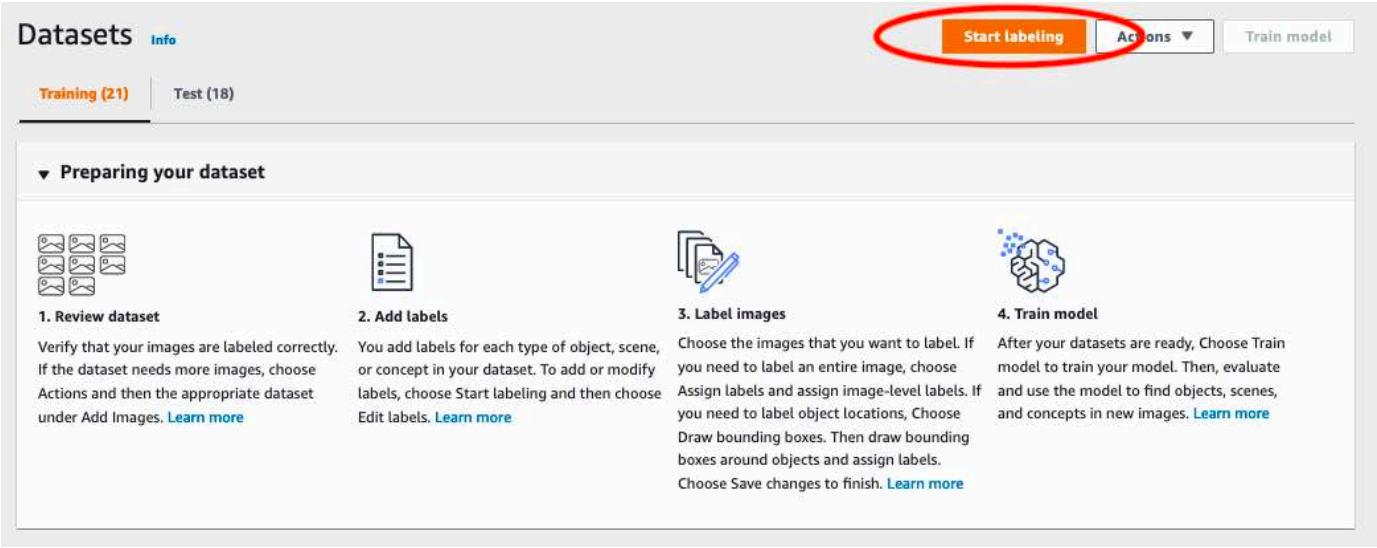
14. 重複步驟 8-12，將影像新增至測試資料集。對於步驟 8，選擇動作然後選擇新增影像至測試資料集。

步驟 5：將標籤添加到項目

在此步驟中，您將標籤添加到項目中為每個在步驟中識別的類步驟 2：決定您的課程。

若要新增標籤 (主控台)

1. 在資料集收藏館頁面上，選擇開始貼標籤進入標籤模式。



2. 在標籤資料集收藏館區段中，選擇編輯標籤以開啟管理標籤對話方塊。
3. 在編輯方塊中，輸入新標示名稱。
4. 選擇加入標籤。
5. 重複步驟 3 和 4，直到建立完所需的所有標籤為止。
6. 選擇儲存以儲存您加入的標示。

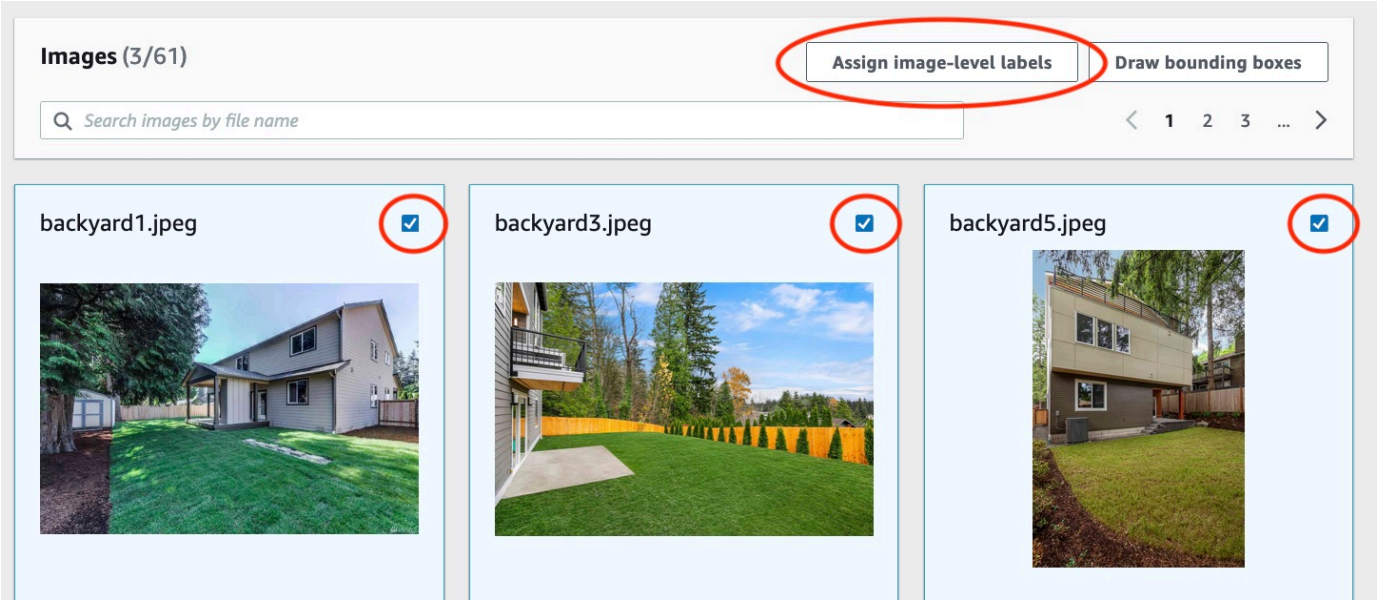
步驟 6：為訓練和測試資料集指派影像層級標籤

在此步驟中，您可以為訓練和測試資料集中的每個映像指派單一映像層級。影像層級標籤是每個影像所代表的類別。

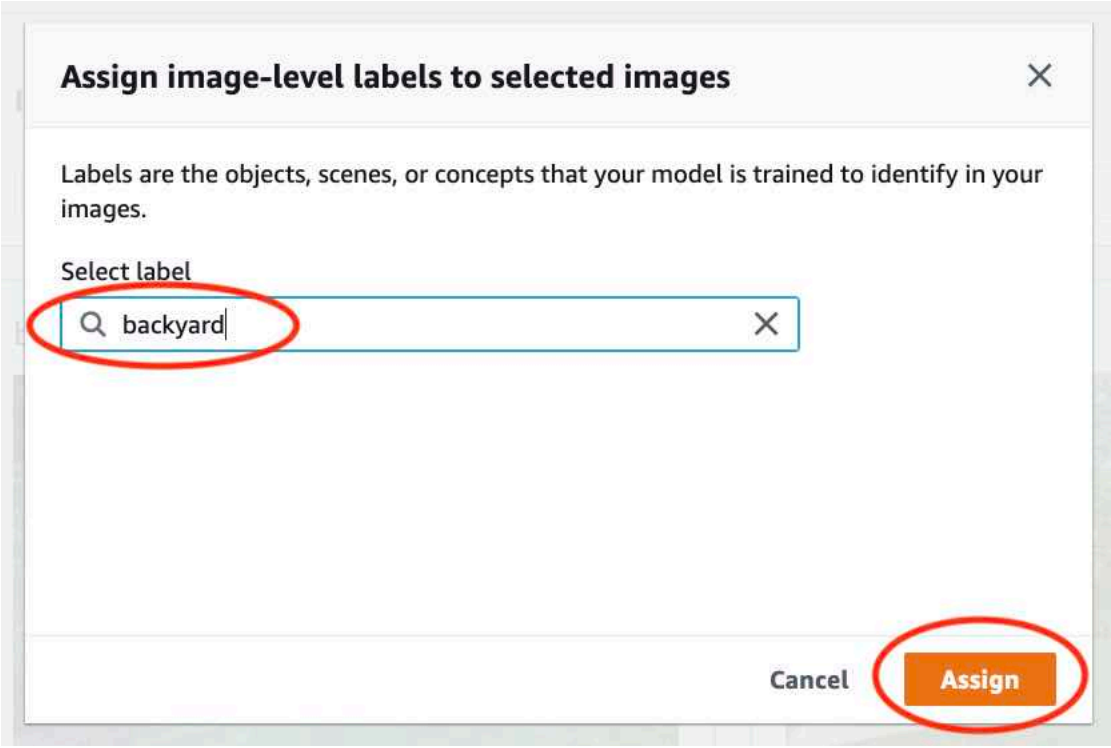
指定影像層級標籤給影像 (主控台)

1. 在「」資料集頁面上，選擇訓練標籤。

2. 選擇開始貼標籤進入標籤模式。
3. 選取您要新增標籤的一或多個影像。您一次只能選取單一頁面上的影像。若要在頁面上選取連續範圍的影像：
   a. 選取第一個影像。
   b. 按住 Shift 鍵。
   c. 選取第二個影像。也會選取第一張與第二個影像之間的影像。
   d. 釋放換檔鍵。
4. 選擇指定影像層級標籤。



5. 在為選取的影像指定影像層級標籤」對話方塊中，選取要指定給一個或多個影像的標籤。
6. 選擇分配以指定影像的標籤。



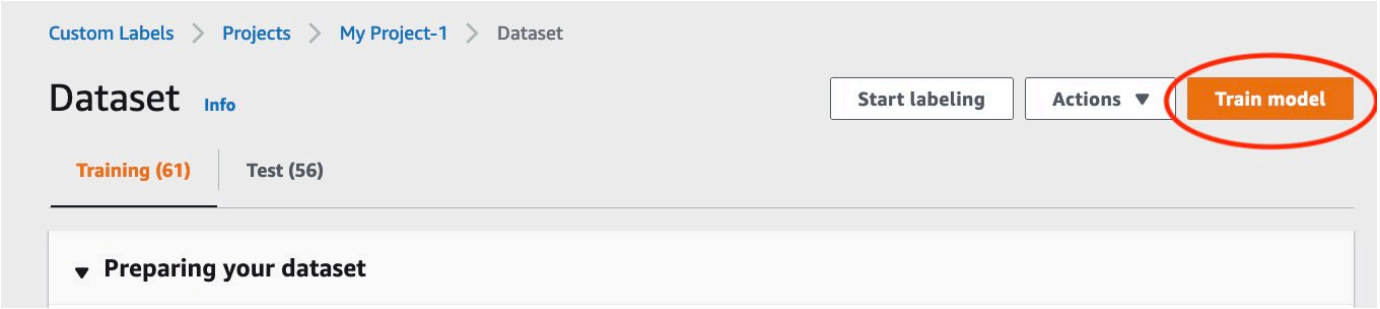
7. 重複標籤，直到每個圖像都用所需的標籤進行註釋。
8. 選擇 測試 標籤。
9. 重複步驟，將影像層級標籤指派給測試資料集影像。

步驟 7：訓練您的模型

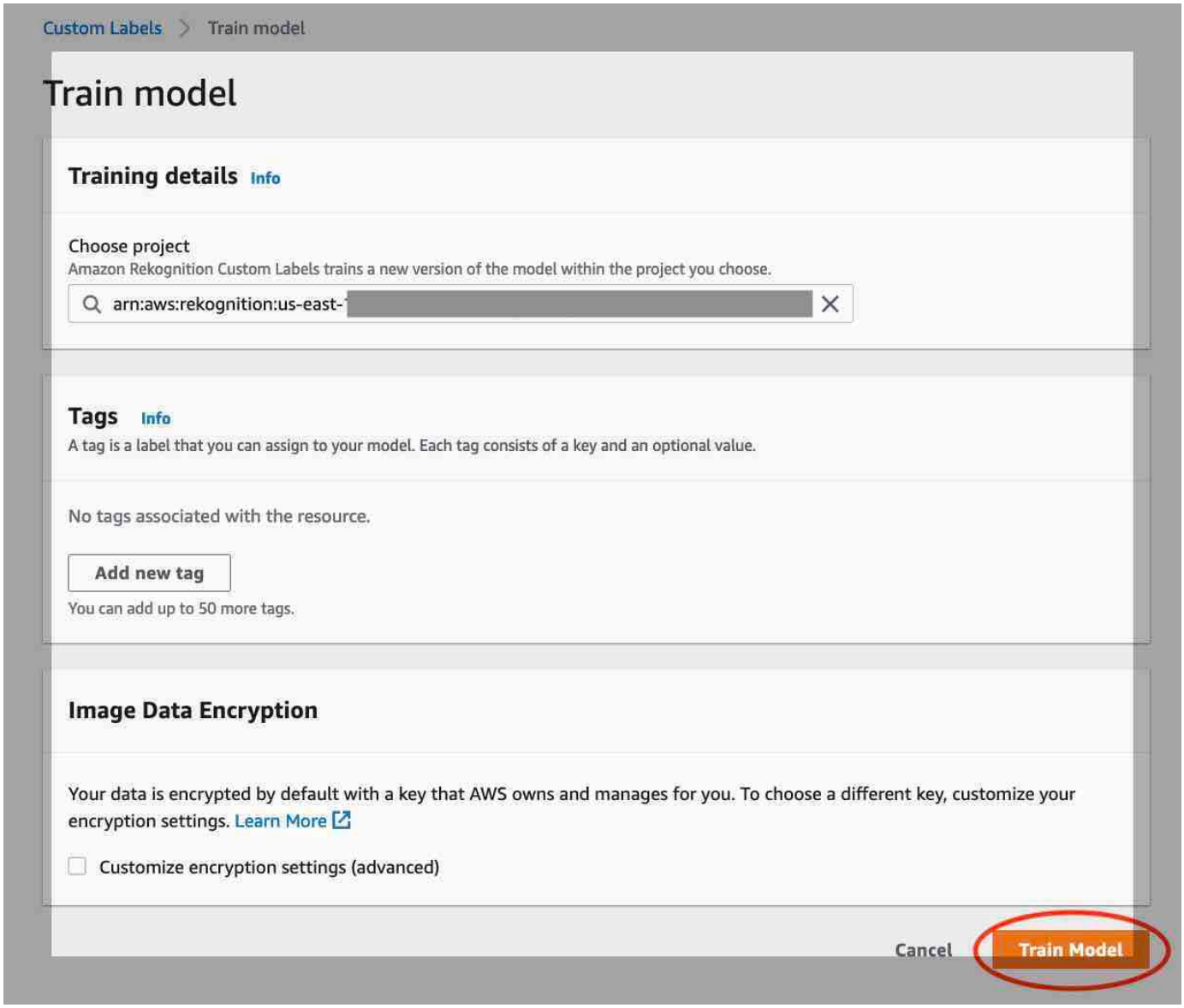
使用下列步驟訓練模型。如需詳細資訊，請參閱訓練 Amazon Rekognition 自訂標籤模型。

訓練您的模型（控制台）

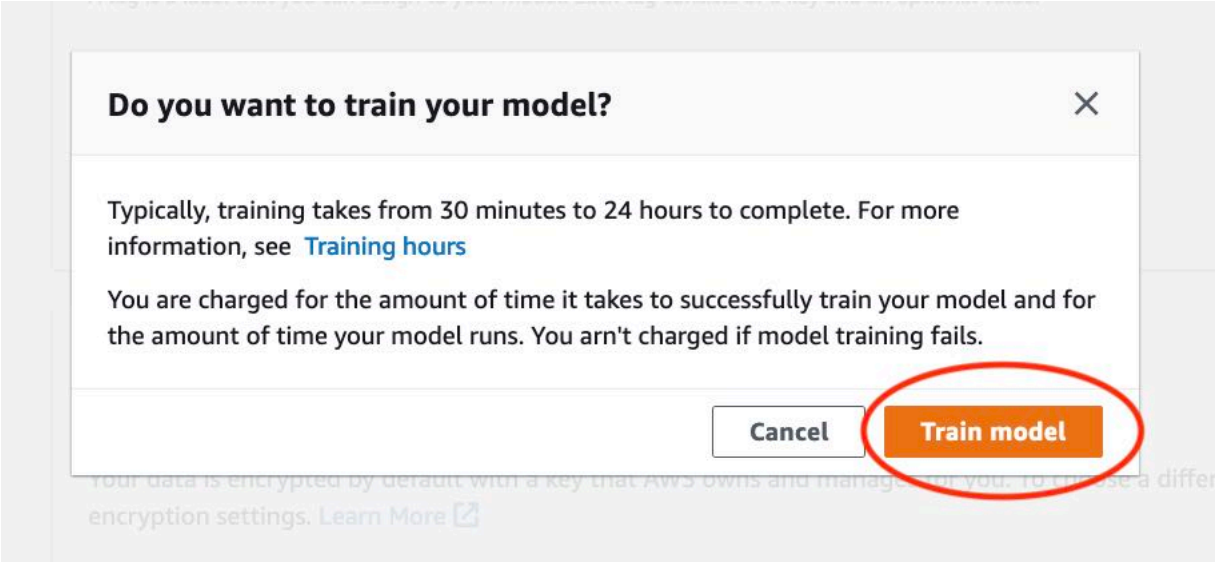
1. 在「」資料集頁面上，選擇火車模型。



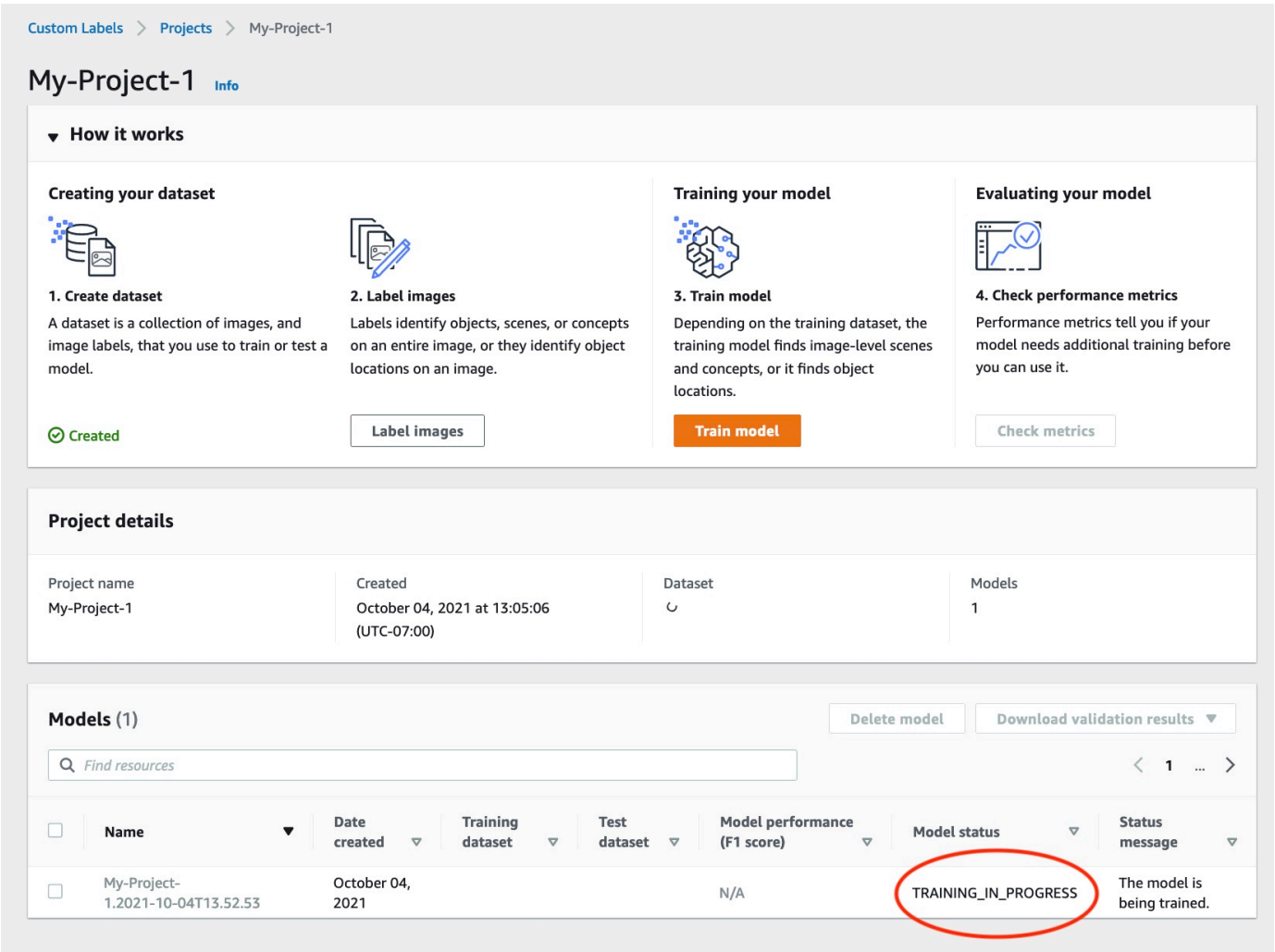
2. 在「」火車模型頁面上，選擇火車模型。您的項目的亞馬遜資源名稱（ARN）位於選擇專案編輯方塊。



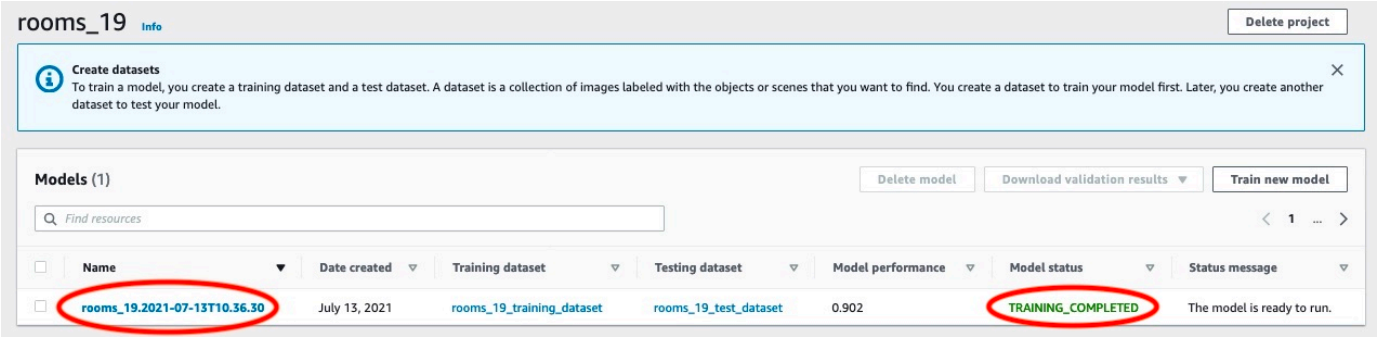
3. 在你想訓練你的模型嗎？」對話方塊中，選擇火車模型。



4. 在模特兒項目頁面的部分，您可以看到培訓正在進行中。您可以檢視目前狀態Model Status模型版本的欄。訓練模型需要一段時間才能完成。

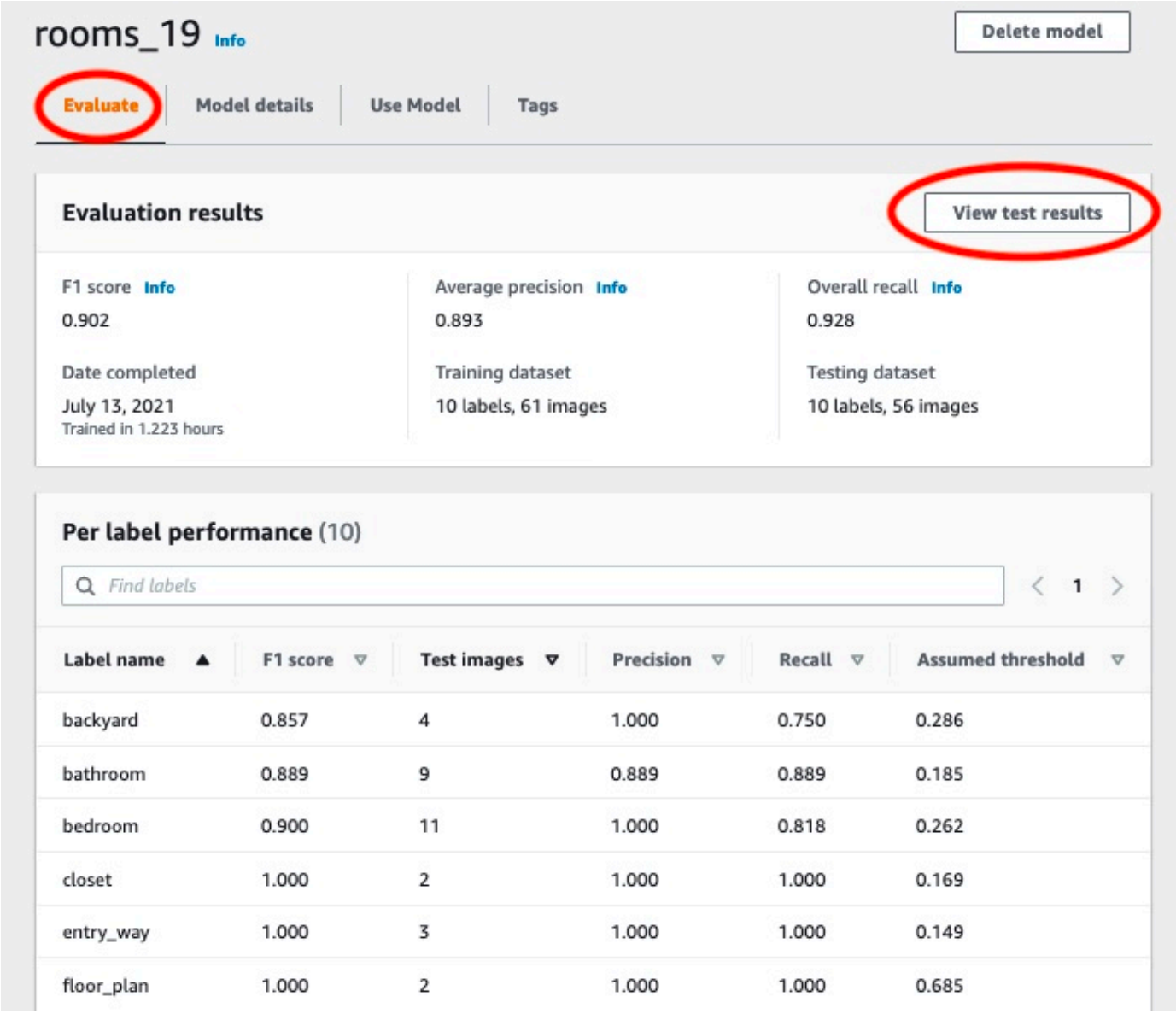


5. 訓練完成後，選擇模型名稱。模型狀態為時，訓練已完成訓練 (_已完成)。

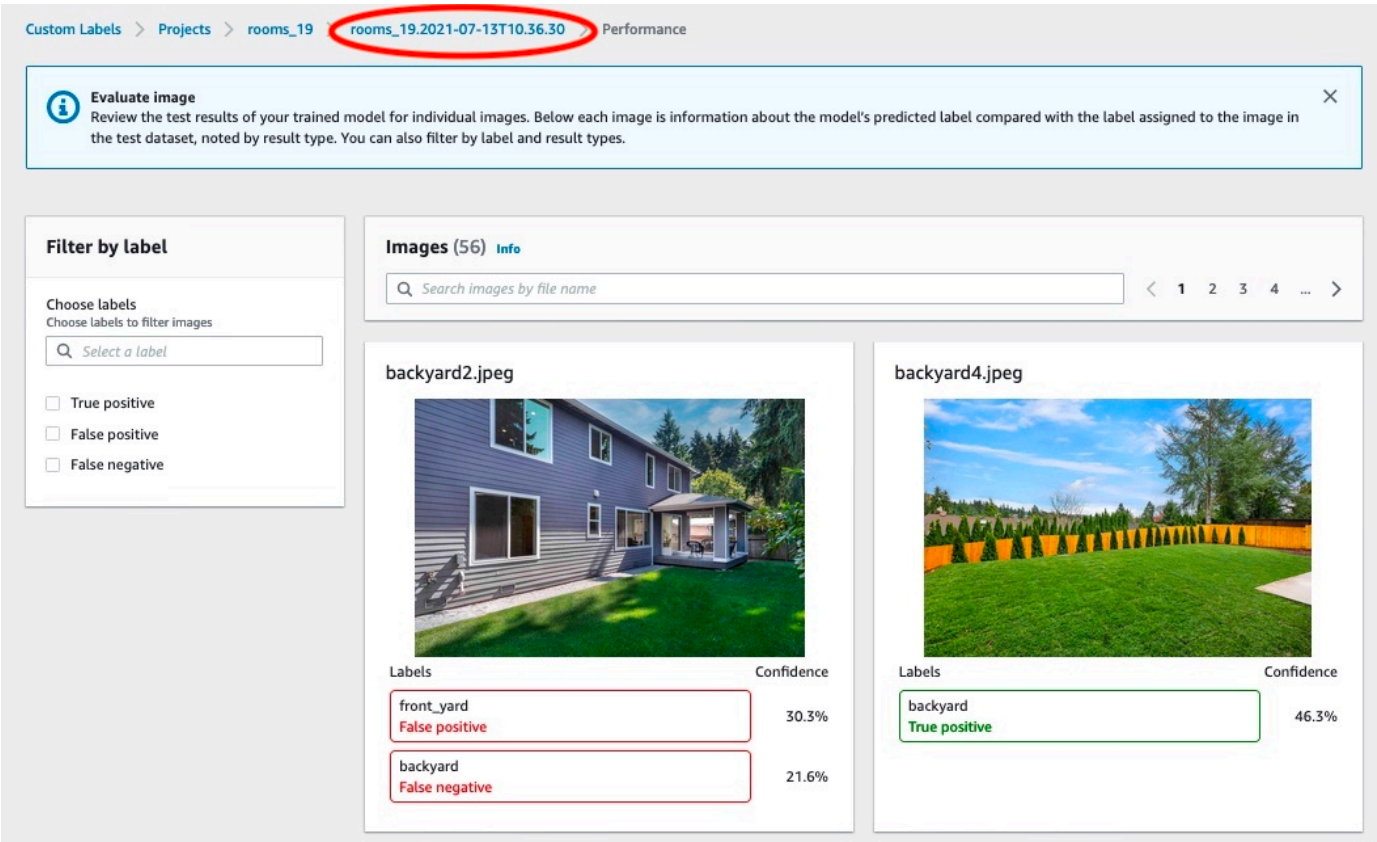


6. 選擇合適的評估按鈕以查看評估結果。如需有關評估模型的資訊，請參閱〈改善訓練有素的 Amazon Rekognition 自訂標籤模型。

7. 選擇檢視測試結果以查看個別測試影像的結果。如需詳細資訊，請參閱評估模型的指標。



8. 檢視測試結果後，選擇要返回模型頁面的型號名稱。



步驟 8：啟動模型

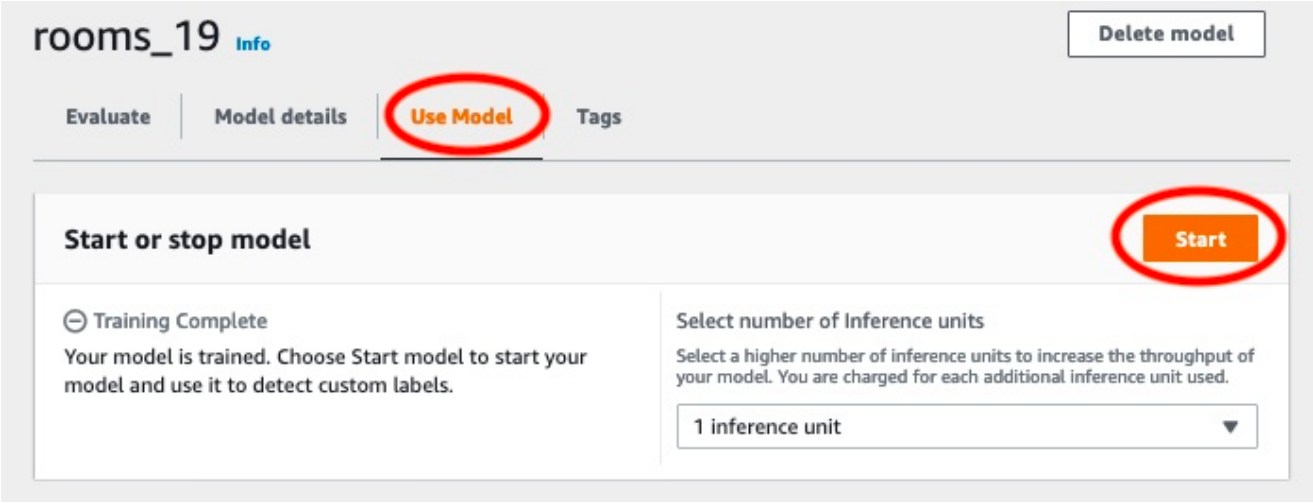
在此步驟中，您可以啟動模型。模型啟動後，您可以使用它來分析影像。

我們會根據模型執行的時間量向您收費。如果您不需要分析影像，請停止模型。您可以稍後重新啟動模型。如需詳細資訊，請參閱執行培訓過的 Amazon Rekognition 自訂標籤模型。

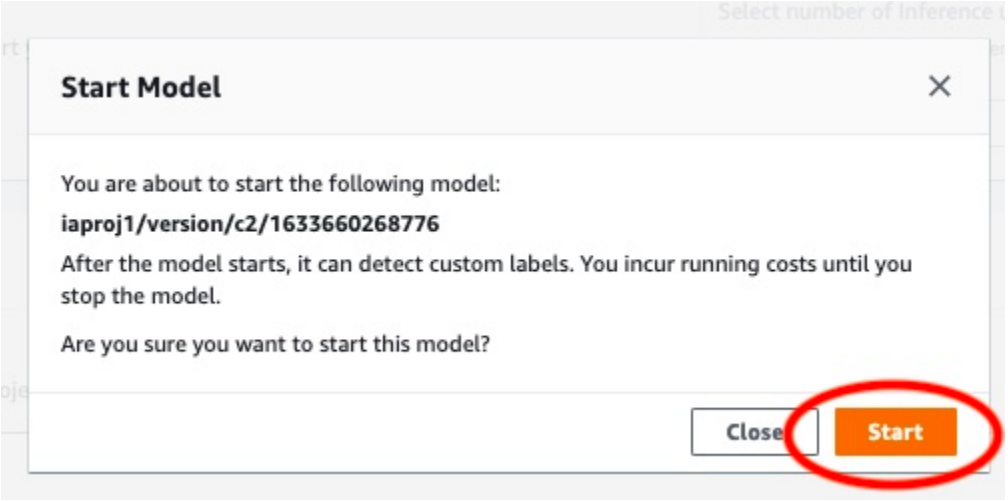
開始您的模型

1. 選擇合適的使用模型模型頁面上的標籤。
2. 在啟動或停止模型區段執行下列動作：

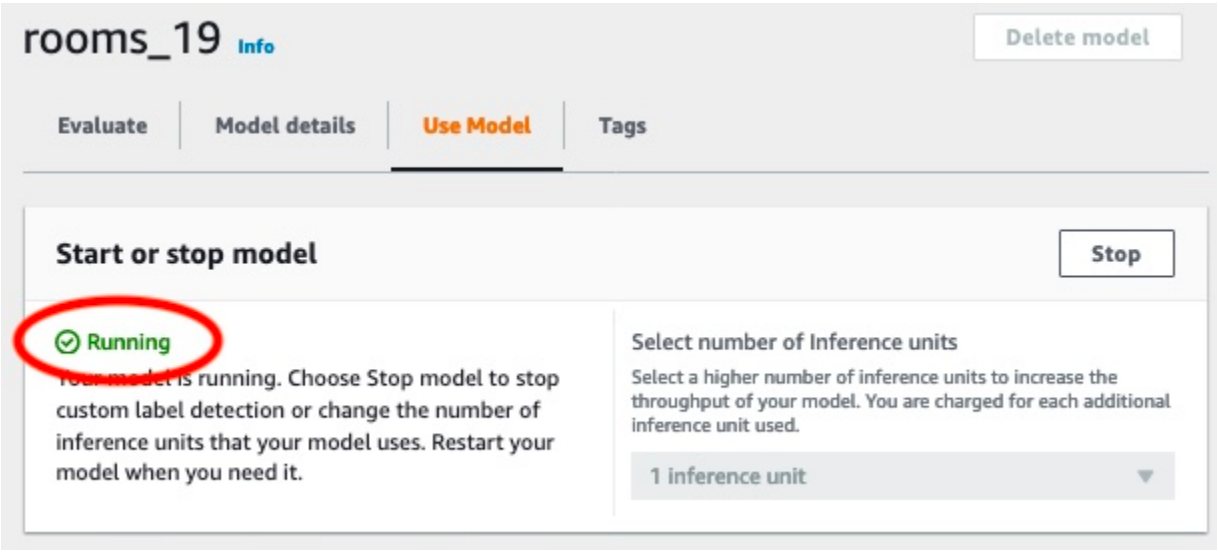
    a. 選擇 Start (啟動)。



b. 在開始模型」對話方塊中，選擇開始。



3. 等到模型運行。模型正在運行時的狀態啟動或停止模型部分是跑步。



步驟 9：使用模型分析圖像

您可以通過調用分析圖像DetectCustomLabelsAPI。在此步驟中，您可以使用`detect-custom-labels` AWS Command Line Interface(AWS CLI) 指令來分析範例影像。你得到AWS CLI來自亞馬遜自訂標籤主控台的命令。控制台配置AWS CLI指令以使用您的模型。您只需要提供存放在 Amazon S3 儲存貯體中的映像。

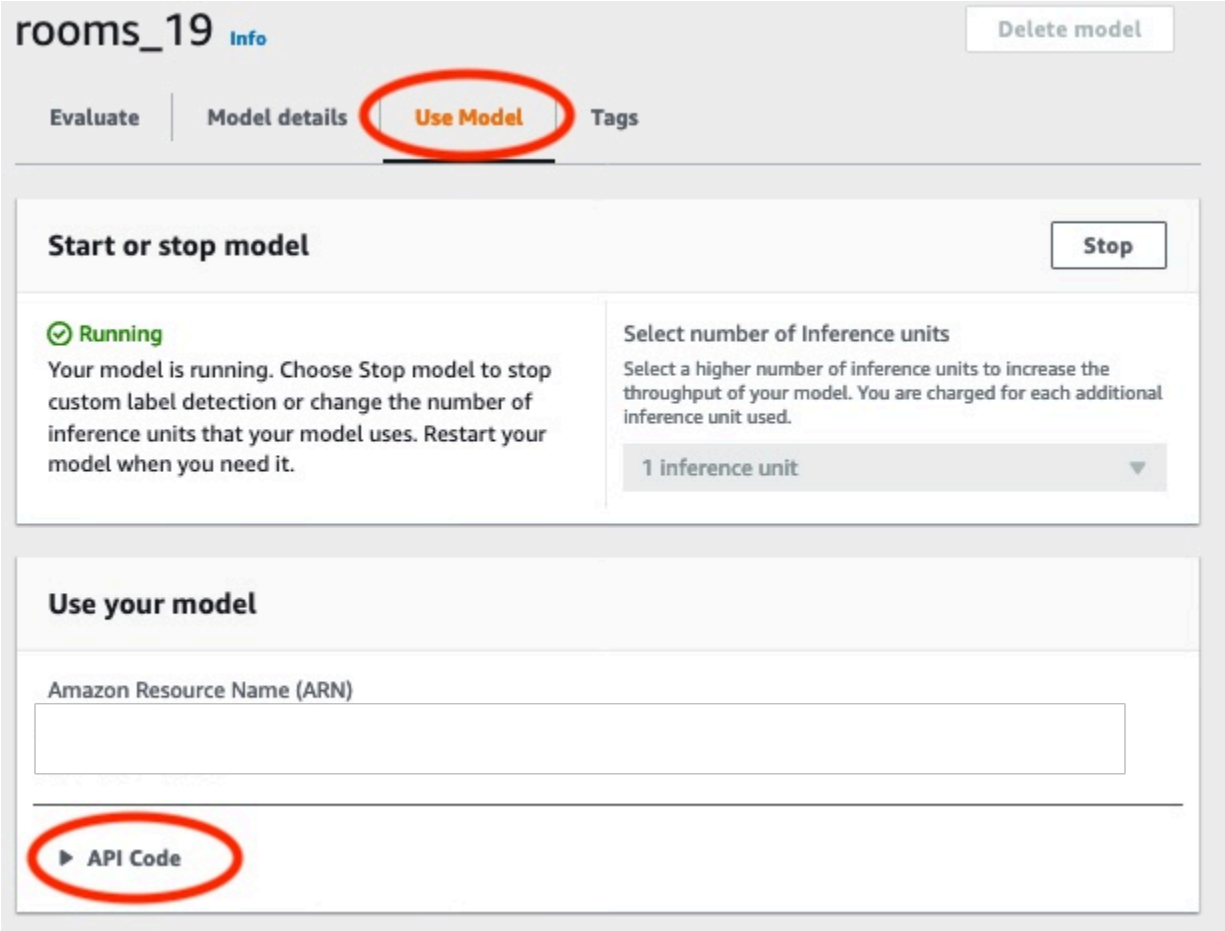
> **Note**
>
> 控制台還提供了 Python 示例代碼。

從輸出`detect-custom-labels`包括在影像中找到的標籤清單、邊界方框 (如果模型找到物件位置)，以及模型對預測準確度的可信度。

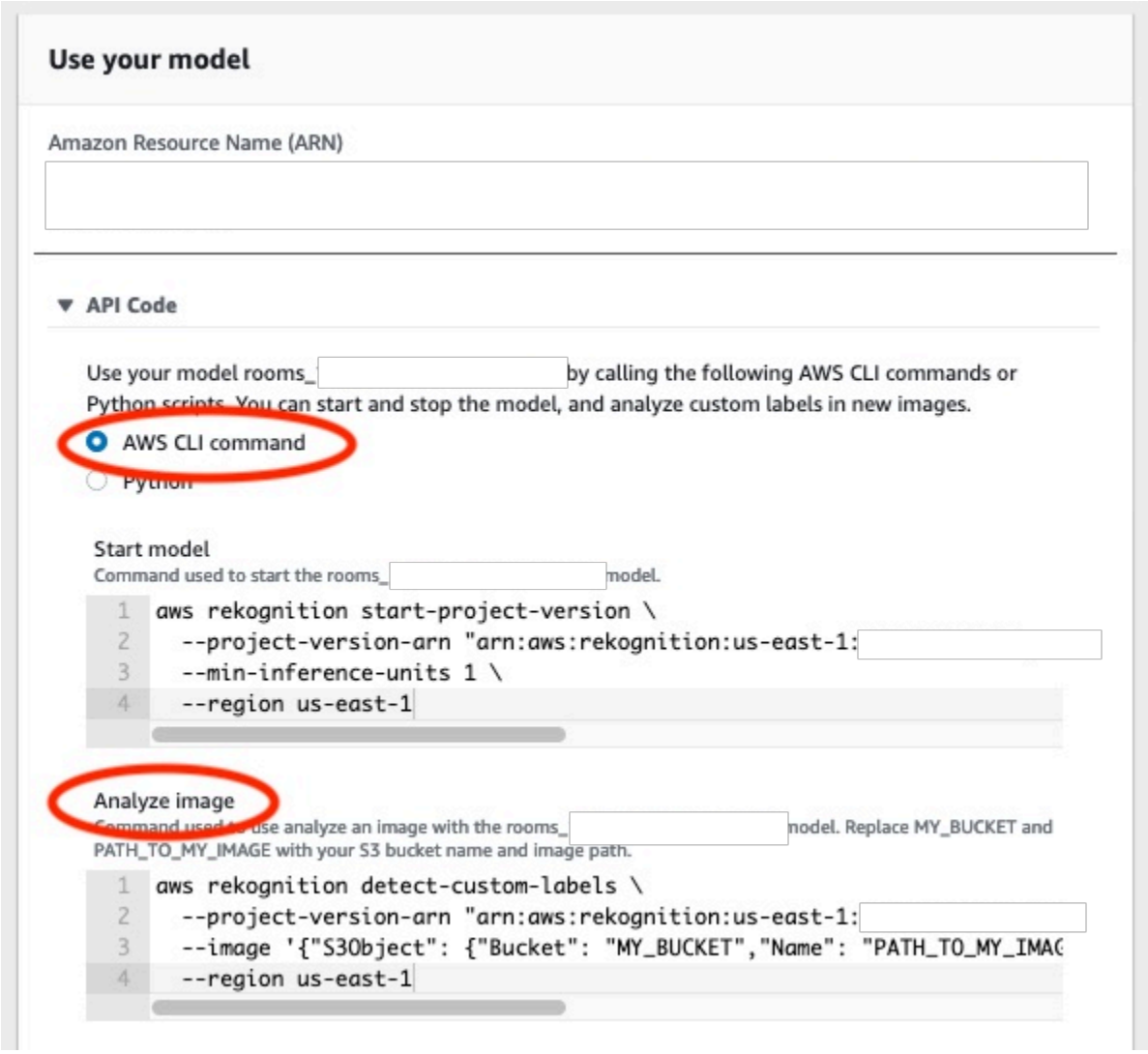
如需詳細資訊，請參閱使用經過培訓的模型分析圖像。

分析影像的步驟 (主控台)

1. 如果您還沒有，請設定AWS CLI。如需相關指示，請參閱the section called “步驟 4：設定 AWS CLI 和 AWS SDKs”。
2. 選擇合適的使用模型標籤，然後選擇API 程式碼。



3. 選擇命令。
4. 在分析影像」區段中，複製AWS CLI呼叫的命令`detect-custom-labels`。



5. 將圖像上傳到亞馬遜 S3 存儲桶。如需指示，請參閱將物件上傳到亞馬遜 S3在亞馬遜簡單存儲服務用戶指南。如果您正在使用 Rooms 專案中的影像，請使用其中一個您移至其他資料夾的影像步驟 1：收集您的圖像。

6. 於指令提示下，輸入AWS CLI您在上一個步驟中複製的指令。它應該看起來像下面的例子。

   的價值`--project-version-arn`應該是您模型的亞馬遜資源名稱（ARN）。的價值`--region`應該是AWS您在其中建立模型的區域。

   變更`MY_BUCKET`和`PATH_TO_MY_IMAGE`到您在上一步中使用的 Amazon S3 儲存貯體和映像檔。

   如果您正在使用custom-labels-access配置文件獲取憑據，添加`--profile custom-labels-access`參數。

```
aws rekognition detect-custom-labels \
  --project-version-arn "model_arn" \
  --image '{"S3Object": {"Bucket": "MY_BUCKET","Name": "PATH_TO_MY_IMAGE"}}' \
  --region us-east-1 \
  --profile custom-labels-access
```

來自的 JSON 輸出AWS CLI命令看起來應該類似於以下內容。Name是模型找到的影像層級標籤名稱。Confidence(0-100) 是模型對預測準確性的信心。

```
{
    "CustomLabels": [
        {
            "Name": "living_space",
            "Confidence": 83.41299819946289
        }
    ]
}
```

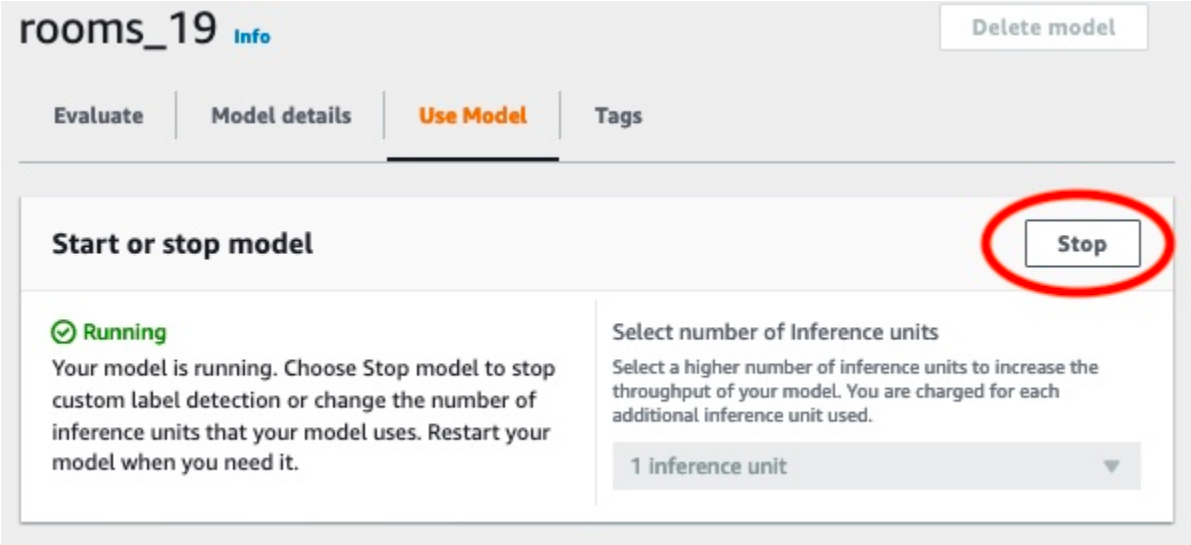
7. 繼續使用模型分析其他影像。如果您不再使用模型，請停止模型。

步驟 10：停止模型

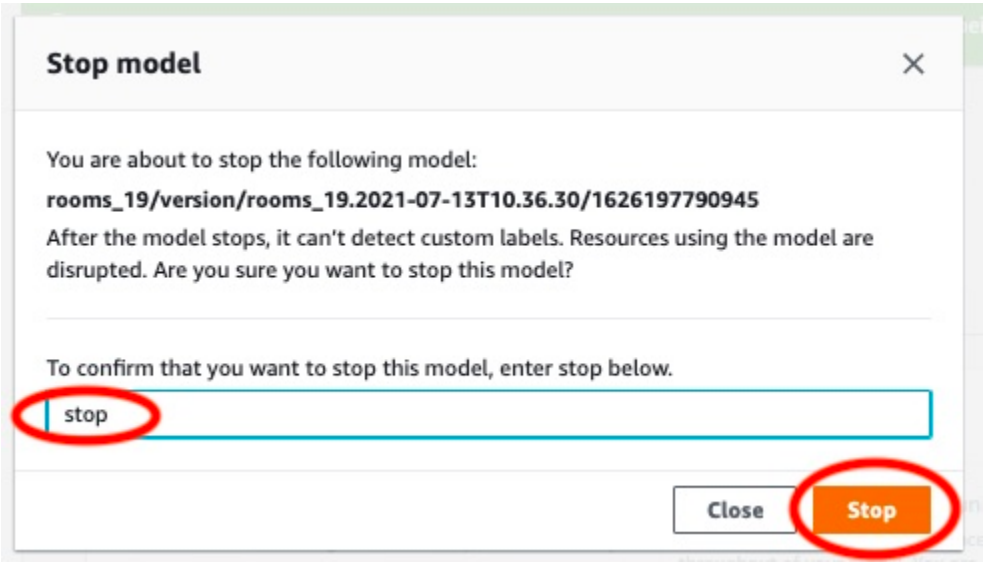
在此步驟中，您將停止執行模型。您需支付模型執行時間的費用。如果您已使用完模型，則應停止模型。

若要停止您的模型

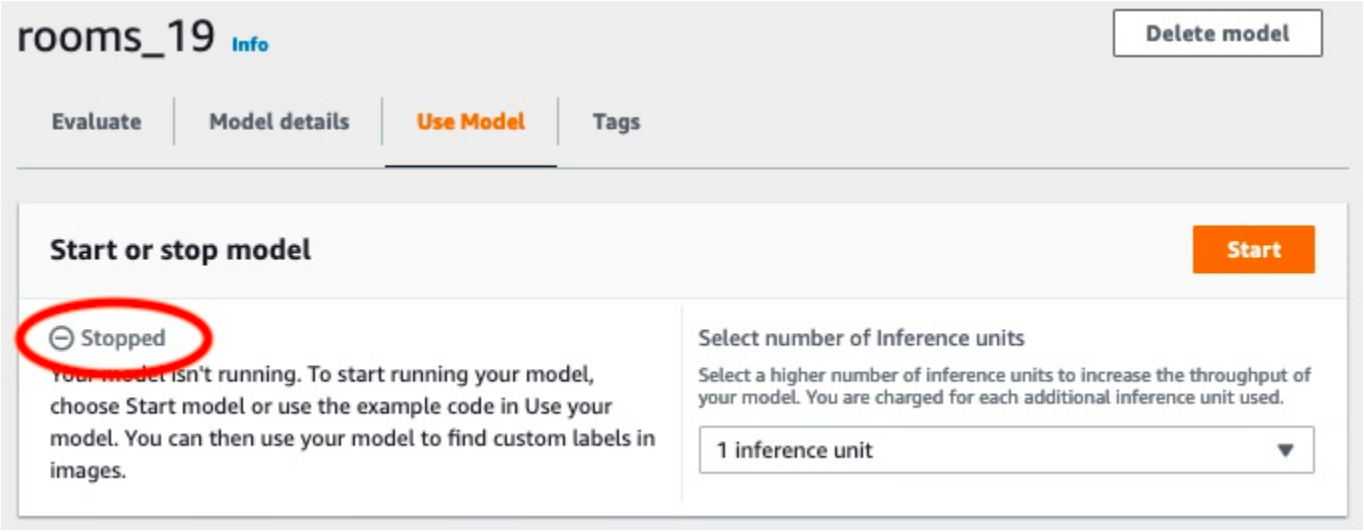
1. 在啟動或停止模型部分選擇停。



2. 在停止模型」對話方塊中，輸入停以確認您要停止模型。



3. 選擇停停止您的模型。當狀態在模型已停止啟動或停止模型部分是已停止。


rooms_19 Info
Delete model
Evaluate
Model details
Use Model
Tags
Start or stop model
Start
Stopped
isn't running. To start running your model, choose Start model or use the example code in Use your model. You can then use your model to find custom labels in images.
Select number of Inference units
Select a higher number of inference units to increase the throughput of your model. You are charged for each additional inference unit used.
1 inference unit

建立 Amazon Rekognition 型

模型是您訓練的軟件，以找到您的業務獨特的概念，場景和對象。您可以使用 Amazon Rekognition 自訂標籤主控台或使用開AWS發套件建立模型。建立 Amazon Rekognition，請先閱讀了解 Amazon Rekognition 自訂標籤。

本節提供有關建立專案、針對不同模型類型建立訓練和測試資料集，以及訓練模型的主控台和 SDK 資訊。後面的章節將向您展示如何改進和使用模型。如需示如何使用主控台建立和使用特定模型的教學，請參閱教學課程：分類影像。

主題

- 建立專案
- 建立訓練和測試資料集
- 訓練 Amazon Rekognition 自訂標籤模型
- 偵錯失敗的模型訓練

建立專案

專案會管理模型版本、訓練資料集和模型的測試資料集。您可以使用 Amazon Rekognition 自訂標籤主控台或 API 在 Amazon Rekognition 自訂標籤主控台中建立專案。如需其他專案任務 (例如刪除專案)，請參閱管理 Amazon Rekognition 自訂標籤專案。

建立 Amazon Rekognition 自訂標籤專案 (主控台)

您可以使用 Amazon Rekognition 自訂標籤主控台來建立專案。當您第一次在新AWS區域中使用主控台時，Amazon Rekognition 自訂標籤會要求在您的AWS帳戶中建立 Amazon S3 儲存貯體 (主控台儲存貯體)。存儲桶用於存儲項目文件。除非建立主控台儲存貯體，否則您無法使用 Amazon Rekognition 自訂標籤主控台。

您可以使用 Amazon Rekognition 自訂標籤主控台來建立專案。

若要建立專案 (主控台)

1. 登入，AWS Management Console並在 https://console.aws.amazon.com/rekognition/ 開啟 Amazon Rekognition 主控台。
2. 在左窗格中，選擇「使用自訂標籤」。顯示 Amazon Rekognition 自訂標籤登陸頁面。
3. 在 Amazon Rekognition 訂標籤登陸頁面中，選擇開始使用。

4. 在左窗格中，選擇專案。
5. 選擇 Create Project (建立專案)。
6. 在 Project name (專案名稱) 中，輸入您的專案名稱。
7. 選擇「創建項目」以創建項目。
8. 請遵循中建立訓練和測試資料集的步驟，為您的專案建立訓練和測試資料集。

建立 Amazon Rekognition 自訂標籤專案 (SDK)

您可以通過致電建立 Amazon Rekognition 自訂標籤專案CreateProject。回應是識別專案的 Amazon Resource Name (ARN)。建立專案後，您可以建立用於訓練和測試模型的資料集。如需詳細資訊，請參閱建立包含影像的訓練和測試資料集。

若要建立專案 (SDK)

1. 若您尚未這樣做，請安裝AWS CLI並設定和AWS SDK。如需詳細資訊，請參閱步驟 4：設定 AWS CLI 和 AWS SDKs。
2. 請使用下面的程式碼來建立專案。

   AWS CLI

   以下範例將建立專案並顯示 ARN。

   `project-name`將的值變更為您要建立專案的名稱。

   ```
   aws rekognition create-project --project-name my_project \
    --profile custom-labels-access
   ```

   Python

   以下範例將建立專案並顯示 ARN。提供下列命令列引數：

   - `project_name`— 要建立專案的名稱。

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

   import argparse
   import logging
   ```

```
import boto3

from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)

def create_project(rek_client, project_name):
    """
    Creates an Amazon Rekognition Custom Labels project
    :param rek_client: The Amazon Rekognition Custom Labels Boto3 client.
    :param project_name: A name for the new prooject.
    """

    try:
        #Create the project.
        logger.info("Creating project: %s",project_name)

        response=rek_client.create_project(ProjectName=project_name)

        logger.info("project ARN: %s",response['ProjectArn'])

        return response['ProjectArn']


    except ClientError as err:
        logger.exception("Couldn't create project - %s: %s", project_name,
 err.response['Error']['Message'])
        raise

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

    parser.add_argument(
        "project_name", help="A name for the new project."
    )


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()

        print(f"Creating project: {args.project_name}")

        # Create the project.
        session = boto3.Session(profile_name='custom-labels-access')
        rekognition_client = session.client("rekognition")

        project_arn=create_project(rekognition_client,
            args.project_name)

        print(f"Finished creating project: {args.project_name}")
        print(f"ARN: {project_arn}")

    except ClientError as err:
        logger.exception("Problem creating project: %s", err)
        print(f"Problem creating project: {err}")


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

Java V2

以下範例將建立專案並顯示 ARN。

提供下列命令列引數：

- project_name— 要建立專案的名稱。

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

```
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.CreateProjectRequest;
import software.amazon.awssdk.services.rekognition.model.CreateProjectResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;

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

public class CreateProject {

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

    public static String createMyProject(RekognitionClient rekClient, String
 projectName) {

        try {

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

            CreateProjectResponse response =
 rekClient.createProject(createProjectRequest);

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

            return response.projectArn();

        } catch (RekognitionException e) {
            logger.log(Level.SEVERE, "Could not create project: {0}",
 e.getMessage());
            throw e;
        }

    }

    public static void main(String[] args) {

        final String USAGE = "\n" + "Usage: " + "<project_name> <bucket> <image>
\n\n" + "Where:\n"
                + "   project_name - A name for the new project\n\n";
```

```
        if (args.length != 1) {
            System.out.println(USAGE);
            System.exit(1);
        }

        String projectName = args[0];
        String projectArn = null;
        ;

        try {

            // Get the Rekognition client.
            RekognitionClient rekClient = RekognitionClient.builder()
                .credentialsProvider(ProfileCredentialsProvider.create("custom-
labels-access"))
                .region(Region.US_WEST_2)
                .build();

            // Create the project
            projectArn = createMyProject(rekClient, projectName);

            System.out.println(String.format("Created project: %s %nProject ARN:
 %s", projectName, projectArn));

            rekClient.close();

        } catch (RekognitionException rekError) {
            logger.log(Level.SEVERE, "Rekognition client error: {0}",
 rekError.getMessage());
            System.exit(1);
        }

    }

}
```

3. 請注意回應中顯示的專案 ARN 名稱。建立模型將會建立模型。
4. 請遵循中建立訓練和測試資料集 (SDK) 的步驟，為您的專案建立訓練和測試資料集。

建立訓練和測試資料集

資料集指描述這些影像的一組影像和標籤。您的專案需要訓練資料集和測試資料集。Amazon Rekognition 自訂標籤會使用訓練資料集來訓練您的模型。訓練結束後，Amazon Rekognition 自訂標籤會使用測試資料集來驗證訓練過的模型預測正確標籤的成效。

您可以使用 Amazon Rekognition 自訂標籤主控台或使用 AWS SDK 建立資料集。建立資料集之前，我們建議您先閱讀 了解 Amazon Rekognition 自訂標籤。如需其他資料集任務，請參閱 管理資料集。

為專案建立訓練和測試資料集的步驟如下：

建立專案的訓練和測試資料集

1. 確定您需要如何標記訓練和測試資料集。如需詳細資訊，規劃資料集。
2. 收集訓練和測試資料集的影像。如需詳細資訊，請參閱 the section called “準備影像”。
3. 建立訓練和測試資料集。如需詳細資訊，請參閱 建立包含影像的訓練和測試資料集。如果您使用 AWS SDK，請參閱 建立訓練和測試資料集 (SDK) 。
4. 如有必要，請新增影像層級標籤或週框方塊至您的資料集影像。如需詳細資訊，請參閱 標記檔案。

建立資料集之後，您即可訓練模型。

主題

規劃資料集

在專案中標記訓練和測試資料集的方式會決定您建立的模型類型。使用 Amazon Rekognition 自訂標籤，您可以建立執行下列動作的模型。

- 尋找物件、場景和概念
- 尋找物件位置
- 尋找品牌位置

尋找物件、場景和概念

模型會將和整個影像相關聯的物件、場景和概念進行分類。

您可以建立兩種類型的分類模型：影像分類和多標籤分類。針對這兩種類型的分類模型，模型會從用於訓練的完整標籤集中尋找一個或多個相符標籤。訓練和測試資料集至少需要兩個標籤。

影像分類

模型會將影像分類為屬於一組預定義的標籤。例如，您可能需要確定影像是否包含居住空間的模型。下列影像可能具有 ling_space 影像層級標籤。

針對此類型的模型，請新增單一影像層級標籤至每個訓練和測試資料集影像。如需範例專案，請參閱影像分類。

多標籤分類

模型會將影像分類為多個類別，例如花卉的類型以及是否有葉子。例如，下列影像可能具有 mediterranean_spurge 和 no_leaves 影像層級標籤。

針對此類型的模型，請指派每個類別的影像層級標籤給訓練和測試資料集影像。如需範例專案，請參閱多標籤圖像分類。

指派影像層級標籤

如果您的影像存放在 Amazon S3 儲存貯體中，您可以使用資料夾名稱自動新增影像層級標籤。如需詳細資訊，請參閱 Amazon S3 儲存貯體。您也可以在建立資料集之後，新增影像層級標籤至影像，如需更多詳細資訊，請參閱 the section called “將影像層級標籤指派給影像”。您可以根據需要新增標籤。如需詳細資訊，請參閱 管理標籤。

尋找物件位置

若要建立預測影像中物件位置的模型，您需要為訓練和測試資料集中的影像定義物件位置週框方塊和標籤。週框方框是緊密圍繞物件的方塊。例如，下列影像即顯示 Amazon Echo 和 Amazon Echo Dot 周圍的週框方塊。每個週框方塊都有一個指派的標籤 (Amazon Echo 或 Amazon Echo Dot)。

若要尋找物件位置，您的資料集至少需要至少一個標籤。在模型訓練期間，會自動建立更多標籤，代表影像上週框方塊外部的區域。

指派週框方塊

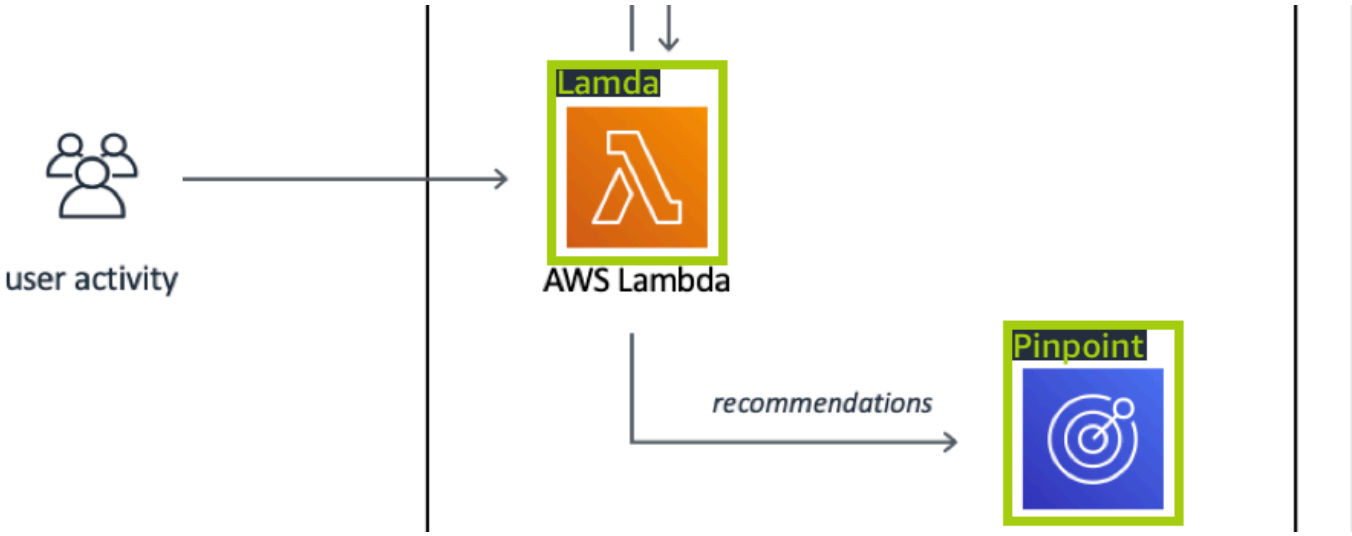
建立資料集時，您可以包含影像的週框方塊資訊。例如，您可以匯入包含邊界方塊的 SageMaker Ground Truth 格式資訊清單檔案。您也可以在建立資料集之後新增週框方塊。如需詳細資訊，請參閱使用週框方塊標記物件。您可以根據需要新增標籤。如需詳細資訊，請參閱 管理標籤。

尋找品牌位置

如果您想要尋找品牌的位置，例如標誌和動畫人物，您可以為訓練資料集影像使用兩種不同類型的影像。

- 僅屬於標誌的影像。每個影像都需要代表標誌名稱的單一影像層級標籤。例如，下列影像的影像層級標籤可能是 Lambda。

- 在自然位置包含標誌的影像，例如足球比賽或建築圖。每個訓練影像都需要圍繞標誌的每個執行個體的週框方塊。例如，下列影像顯示的架構圖表包含圍繞 AWS Lambda 和 Amazon Pinpoint 標誌的已標記週框方塊。



我們建議您不要在訓練影像中混合使用影像層級的標籤和週框方塊。

測試影像必須在您要尋找的品牌執行個體周圍有週框方塊。只有在訓練影像包含標記的週框方塊時，您才可以分割訓練資料集來建立測試資料集。如果訓練影像只具有影像層級標籤，您即必須建立測試資料集，其中需包含影像和標記的週框方塊。如果您訓練模型來尋找品牌位置，請根據您標記影像的方式進行 使用週框方塊標記物件 和 將影像層級標籤指派給影像。

品牌檢測 範例專案會顯示 Amazon Rekognition 自訂標籤如何使用標記的週框方塊來訓練尋找物件位置的模型。

模型類型的標籤需求

使用下表確定如何標記影像。

您可以在單一資料集中合併影像層級標籤和已標記影像的週框方塊。在這種情況下，Amazon Rekognition 自訂標籤會選擇要建立影像層級模型或是物件位置模型。

範例	訓練影像	測試影像
影像分類	每個影像 1 個影像層級標籤	每個影像 1 個影像層級標籤
多標籤分類	每個影像多個影像層級標籤	每個影像多個影像層級標籤
尋找品牌位置	影像層級標籤 (您也可以使用標記的週框方塊)	標記的週框方塊
尋找物件位置	標記的週框方塊	標記的週框方塊

準備影像

訓練和測試資料集中的影像包含您希望模型尋找的物件、場景或概念。

影像的內容應具有各種背景和光源，以代表您希望訓練過的模型識別的影像。

本區段會提供訓練和測試資料集中影像的相關資訊。

影像格式

您可以使用 PNG 和 JPEG 格式的影像來訓練 Amazon Rekognition 自訂標籤。同樣地，若要使用 `DetectCustomLabels` 偵測自訂標籤，您需要 PNG 和 JPEG 格式的影像。

輸入影像建議

Amazon Rekognition 自訂標籤需要影像來訓練和測試您的模型。為了準備影像，請考慮下列事項：

- 針對您要建立的模型選擇特定網域。例如，您可以為風景檢視選擇模型，為諸如機器零件之類的物件選擇另一個模型。如果您的影像位於所選網域中，Amazon Rekognition 自訂標籤的成效最佳。
- 至少使用 10 個影像來訓練您的模型。
- 影像必須採用 PNG 或 JPEG 格式。
- 使用以各種光源、背景和解析度顯示物件的影像。

- 訓練和測試影像應與您要和模型搭配使用的影像類似。
- 決定將那些標籤指派給影像。
- 確保影像的解析度足夠大。如需詳細資訊，請參閱 Amazon Rekognition 自訂標籤中的準則。
- 確保遮蔽物不會遮掩您要偵測的物件。
- 使用與背景充分對比的映像。
- 使用明亮且銳利的映像。盡量避免使用可能會因主體和相機移動而模糊的影像。
- 使用物件佔據影像很大比例的影像。
- 測試資料集中的影像不應該是訓練資料集中的影像。它們應該包括訓練模型來分析的物件、場景和概念。

影像集大小

Amazon Rekognition 自訂標籤會使用一組影像來訓練模型。最起碼，您應該至少使用 10 個影像進行訓練。Amazon Rekognition 自訂標籤會在資料集中存放訓練和測試影像。如需詳細資訊，請參閱 建立包含影像的訓練和測試資料集。

建立包含影像的訓練和測試資料集

您可以從具有單一資料集的專案開始，或具有不同訓練和測試資料集的專案開始。如果您從單一資料集開始，Amazon Rekognition 自訂標籤會在訓練期間分割您的資料集，以為您的專案建立訓練資料集 (80%) 和測試資料集 (%20)。如果您希望 Amazon Rekognition 自訂標籤決定影像用於訓練和測試的位置，請從單一資料集開始。為了完全控制訓練、測試和效能調校，我們建議您使用不同的訓練和測試資料集來啟動專案。

從下列其中一個位置匯入影像，即可為專案建立訓練和測試資料集：

- Amazon S3 儲存貯體
- 本機電腦
- 清單檔案
- 現有的資料集

如果您使用不同的訓練和測試資料集啟動專案，即可針對每個資料集使用不同的來源位置。

依據您匯入影像的位置而定，您的影像可能沒有標記。例如，從本機電腦匯入的圖像不會加上標籤。從 Amazon SageMaker Ground Truth 資訊清單檔案匯入的影像會加上標籤。您可以使用 Amazon Rekognition 自訂標籤主控台來新增、變更和分配標籤。如需詳細資訊，請參閱 標記檔案。

如果上傳的影像有錯誤、影像遺失或影像缺少標籤，請閱讀 偵錯失敗的模型訓練。

如需資料集的詳細資訊，請參閱 管理資料集。

建立訓練和測試資料集 (SDK)

您可以使用 AWS SDK 建立訓練和測試資料集。

訓練資料集

您可用下列方式使用 AWS SDK 來建立訓練資料集。

- 搭CreateDataset配您提供的 Amazon Sageemaker 格式資訊清單檔案搭配使用。如需詳細資訊，請參閱 the section called “建立清單檔案”。如需範例程式碼，請參閱 使用 SageMaker Ground Truth 資訊清單檔案 (SDK) 建立資料集。
- 使用 `CreateDataset` 複製現有的 Amazon Rekognition 自訂標籤資料集。如需範例程式碼，請參閱 使用現有資料集 (SDK) 建立資料集。
- 使用 `CreateDataset` 建立空白資料集，並在稍後使用 UpdateDatasetEntries 新增資料集項目。若要建立空白資料集，請參閱 將資料集新增至專案。若要新增影像至資料集，請參閱 新增更多影像 (SDK)。您需要先新增資料集項目，才能訓練模型。

測試資料集

您可用下列方式使用 AWS SDK 來建立測試資料集：

- 搭CreateDataset配您提供的 Amazon Sageemaker 格式資訊清單檔案搭配使用。如需詳細資訊，請參閱 the section called “建立清單檔案”。如需範例程式碼，請參閱 使用 SageMaker Ground Truth 資訊清單檔案 (SDK) 建立資料集。
- 使用 `CreateDataset` 複製現有的 Amazon Rekognition 自訂標籤資料集。如需範例程式碼，請參閱 使用現有資料集 (SDK) 建立資料集。
- 使用 `CreateDataset` 建立空白資料集，並在稍後使用 `UpdateDatasetEntries` 新增資料集項目。若要建立空白資料集，請參閱 將資料集新增至專案。若要新增影像至資料集，請參閱 新增更多影像 (SDK)。您需要先新增資料集項目，才能訓練模型。
- 將訓練資料集分割為不同的訓練和測試資料集。首先使用 `CreateDataset` 建立空白測試資料集。接著，透過呼叫，將 20% 的訓練資料集項目移至測試資料集DistributeDatasetEntries。若要建立空白資料集，請參閱 將資料集新增至專案 (SDK)。若要分割訓練資料集，請參閱發佈訓練資料集 (SDK)。

Amazon S3 儲存貯體

從 Amazon S3 儲存貯體會入影像。您可以使用主控台儲存貯體或 AWS 帳戶中的其他 Amazon S3 儲存貯體。如果您正在使用主控台儲存貯體，則已設定所需權限。如果您沒有使用主控台儲存貯體，請參閱 存取外部 Amazon S3 儲存貯體。

> **Note**
>
> 您無法使用 AWS SDK 直接從 Amazon S3 儲存貯體中的影像建立資料集。請改為建立參考影像來源位置的清單檔案。如需更多資訊，請參閱 清單檔案

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，為影像指派標籤名稱。資料夾必須是您在資料集建立期間在 S3 資料夾位置中指定的 Amazon S3 資料夾路徑的子系。若要建立資料集，請參閱 從 S3 儲存貯體匯入影像以建立資料集。

例如，假設 Amazon S3 儲存貯體具有下列資料夾結構。如果您將 Amazon S3 資料夾位置指定為 S3-bucket/alexa-devices，標籤 echo 即會被指派給資料夾 echo 中的影像。同樣地，標籤 echo-dot 會被指派給資料夾 echo-dot 中的影像。較深的子資料夾的名稱不會用於標記影像。而會改用 Amazon S3 資料夾位置的適當子資料夾。例如，將資料夾中的影像指white-echo-dots定為回波點標籤。S3 資料夾位置 (alexa-devices) 層級的影像沒有被指派的標籤。

指定較深的 S3 資料夾位置，即可使用資料夾結構中較深的資料夾來標記影像。例如，如果您指定 S3-儲存區/Alexa-裝置/迴聲點，則會標示資料夾中的影像。white-echo-dotwhite-echo-dot不會匯入指定的 s3 資料夾位置以外的影像，例如 echo。

```
S3-bucket
### alexa-devices
    ### echo
    #   ### echo-image-1.png
    #   ### echo-image-2.png
    #   ### .
    #   ### .
    ### echo-dot
        ### white-echo-dot
        #   ### white-echo-dot-image-1.png
        #   ### white-echo-dot-image-2.png
        #
        ### echo-dot-image-1.png
        ### echo-dot-image-2.png
        ### .
```

```
    ### .
```

我們建議您在目前的 AWS 區域第一次開啟主控台時，使用 Amazon Rekognition 為您建立的 Amazon S3 儲存貯體 (主控台儲存貯體)。如果您正在使用的 Amazon S3 儲存貯體與主控台儲存貯體不同 (外部)，則主控台會在建立資料集期間提示您設定適當的權限。如需詳細資訊，請參閱 the section called “步驟 2：設定主控台權限”。

從 S3 儲存貯體匯入影像以建立資料集

下列程序會說明如何使用存放在 Console S3 儲存貯體中的影像來建立資料集。影像會自動以存放影像的資料夾名稱標記。

匯入影像之後，您即可從資料集圖庫頁面新增更多影像、指派標籤，以及新增週框方塊。如需詳細資訊，請參閱 標記檔案。

將您的影像上傳到 Amazon Simple Storage Service (S3) 儲存貯體

1. 在本機檔案系統上建立資料夾。使用資料夾名稱，例如 alexa-devices。
2. 在您剛建立的資料夾中，建立以您要使用的每個標籤命名的資料夾。例如，echo 和 echo-dot。資料夾結構應該類似下列內容。

   ```
   alexa-devices
   ### echo
   #   ### echo-image-1.png
   #   ### echo-image-2.png
   #   ### .
   #   ### .
   ### echo-dot
       ### echo-dot-image-1.png
       ### echo-dot-image-2.png
       ### .
       ### .
   ```

3. 將與標籤對應的影像放入具有相同標籤名稱的資料夾中。
4. 登入 AWS Management Console，並開啟位於 https://console.aws.amazon.com/s3/ 的 Amazon S3 主控台。
5. 在第一次設定期間，將您在步驟 1 中建立的資料夾新增至 Amazon Rekognition 自訂標籤為您建立的 Amazon S3 儲存貯體 (主控台儲存貯體)。如需詳細資訊，請參閱 管理 Amazon Rekognition 自訂標籤專案。

6. 在 https://console.aws.amazon.com/rekognition/ 開啟 Amazon Rekognition 主控台。
7. 選擇使用自訂標籤。
8. 選擇 啟動。
9. 在左側導覽窗格中，選擇專案。
10. 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。
11. 選擇建立資料集。建立資料集頁面即會顯示。
12. 在開始設定中，選擇從單一資料集開始或從訓練資料集開始。若要建立更高品質的模型，我們建議您從個別的訓練和測試資料集開始。

    Single dataset

    a. 在訓練資料集詳細資訊區段中，選擇從 S3 儲存貯體匯入影像。
    b. 在訓練資料集詳細資訊區段的影像來源設定區段中，輸入步驟 13 至 15 的資訊。

    Separate training and test datasets

    a. 在訓練資料集詳細資訊區段中，選擇從 S3 儲存貯體匯入影像。
    b. 在訓練資料集詳細資訊區段的影像來源設定區段中，輸入步驟 13 至 15 的資訊。
    c. 在測試資料集詳細資訊區段中，選擇從 S3 儲存貯體匯入影像。
    d. 在測試資料集詳細資訊區段的影像來源設定區段中，輸入步驟 13 至 15 的資訊。
13. 選擇從 Amazon S3 儲存貯體匯入影像。
14. 在 S3 URI 中，輸入 Amazon S3 儲存貯體位置和資料夾路徑。
15. 選擇根據資料夾自動將標籤連接至影像。
16. 選擇建立資料集。專案的資料集頁面隨即開啟。
17. 如果您需要新增或變更標籤，請執行 標記檔案。
18. 請遵循 訓練模型（控制台） 中的步驟訓練模型。

本機電腦

影像會直接從您的電腦載入。您一次最多可以上傳 30 個影像。

您上傳的影像不會有與其相關連的標籤。如需詳細資訊，請參閱 標記檔案。如果您要上傳許多影像，請考慮使用 Amazon S3 儲存貯體。如需詳細資訊，請參閱 Amazon S3 儲存貯體。

> **Note**
>
> 您無法使用 AWS SDK 建立包含本機影像的資料集。請改為建立清單檔案，並將影像上傳至 Amazon S3 儲存貯體。如需詳細資訊，請參閱 清單檔案。

使用本機電腦 (主控台) 上的影像建立資料集

1. 在 https://console.aws.amazon.com/rekognition/ 開啟 Amazon Rekognition 主控台。
2. 選擇使用自訂標籤。
3. 選擇 啟動。
4. 在左側導覽窗格中，選擇專案。
5. 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。
6. 選擇建立資料集。建立資料集頁面即會顯示。
7. 在開始設定中，選擇從單一資料集開始或從訓練資料集開始。若要建立更高品質的模型，我們建議您從個別的訓練和測試資料集開始。

   Single dataset

   a. 在訓練資料集詳細資訊區段中，選擇從您的電腦上傳影像。
   b. 選擇建立資料集。
   c. 在專案的資料集頁面上，選擇新增影像。
   d. 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。
   e. 選擇上傳影像。

   Separate training and test datasets

   a. 在訓練資料集詳細資訊區段中，選擇從您的電腦上傳影像。
   b. 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。

c. 選擇建立資料集。專案的資料集頁面隨即顯示，其中會包含各自資料集的訓練索引標籤和測試索引標籤。

d. 選擇動作，然後選擇新增影像至訓練資料集。

e. 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。

g. 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。

8. 請遵循 標記檔案 中的步驟標記影像。
9. 請遵循 訓練模型（控制台） 中的步驟訓練模型。

清單檔案

您可以使用 Amazon SageMaker Ground Truth 格式資訊清單檔案建立資料集。您可以使用 Amazon SageMaker Ground Truth 任務中的清單文件。如果您的影像和標籤不是 SageMaker Ground Truth 資訊清單檔案的格式，您可以建立 SageMaker 格式資訊清單檔案，並使用它來匯入您的標籤影像。

主題

- 使用 SageMaker Ground Truth 清單文件創建數據集（控制台）
- 使用 SageMaker Ground Truth 資訊清單檔案 (SDK) 建立資料集
- Amazon SageMaker Ground Truth 工作
- 建立清單檔案
- 將其他資料集格式轉換為清單檔案

使用 SageMaker Ground Truth 清單文件創建數據集（控制台）

下列程序說明如何使用 SageMaker Ground Truth 格式資訊清單檔案建立資料集。

1. 執行下列其中一項操作，建立訓練資料集的清單檔案：

   - 依照中的指示建立包含 SageMaker GroundTruth Job 的資訊清單檔案Amazon SageMaker Ground Truth 工作。
   - 依照 建立清單檔案 中的指示，建立您自己的清單檔案。

   如果您要建立測試資料集，請重複步驟 1 即可建立測試資料集。

2. 在 https://console.aws.amazon.com/rekognition/ 開啟 Amazon Rekognition 主控台。

3. 選擇使用自訂標籤。
4. 選擇 啟動。
5. 在左側導覽窗格中，選擇專案。
6. 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。
7. 選擇建立資料集。建立資料集頁面即會顯示。
8. 在開始設定中，選擇從單一資料集開始或從訓練資料集開始。若要建立更高品質的模型，我們建議您從個別的訓練和測試資料集開始。

   Single dataset

   a. 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
   b. 在 .manifest 檔案位置，輸入您在步驟 1 建立之清單檔案的位置。
   c. 選擇建立資料集。專案的資料集頁面隨即開啟。

   Separate training and test datasets

   a. 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
   b. 在 .manifest 檔案位置，輸入您在步驟 1 建立之訓練資料集清單檔案的位置。
   c. 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

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

   d. 在 .manifest 檔案位置，輸入您在步驟 1 建立之測試資料集清單檔案的位置。
   e. 選擇建立資料集。專案的資料集頁面隨即開啟。
9. 如果您需要新增或變更標籤，請執行 標記檔案。
10. 請遵循 訓練模型（控制台） 中的步驟訓練模型。

使用 SageMaker Ground Truth 資訊清單檔案 (SDK) 建立資料集

下列程序說明如何使用 CreateDatasetAPI 從資訊清單檔案建立訓練或測試資料集。

您可以使用現有的資訊清單檔案，例如 SageMaker Ground Truth 工作的輸出，或建立您自己的資訊清單檔案。

1. 如果您尚未安裝和設定 AWS CLI 和 AWS SDK，請先完成此步驟。如需詳細資訊，請參閱 步驟 4：設定 AWS CLI 和 AWS SDKs。
2. 執行下列其中一項操作，建立訓練資料集的清單檔案：

   - 依照中的指示建立包含 SageMaker GroundTruth Job 的資訊清單檔案Amazon SageMaker Ground Truth 工作。
   - 依照 建立清單檔案 中的指示，建立您自己的清單檔案。

   如果您要建立測試資料集，請重複步驟 2 即可建立測試資料集。
3. 使用下列程式碼範例建立訓練和測試資料集。

   AWS CLI

   使用下列程式碼建立資料集。取代以下項目：

   - `project_arn` — 您要為其新增測試資料集的專案的 ARN。
   - `type`— 您要建立的資料集類型 (訓練或測試)
   - `bucket` - 包含資料集之清單檔案的儲存貯體。
   - `manifest_file` - 清單檔案的路徑和檔案名稱

```
aws rekognition create-dataset --project-arn project_arn \
  --dataset-type type \
  --dataset-source '{ "GroundTruthManifest": { "S3Object": { "Bucket": "bucket",
 "Name": "manifest_file" } } }' \
  --profile custom-labels-access
```

   Python

   使用下列值建立資料集。提供下列命令列參數：

   - `project_arn` — 您要為其新增測試資料集之專案的 ARN。
   - `dataset_type` — 您要建立的資料集類型 (`train` 或 `test`)。
   - `bucket` - 包含資料集之清單檔案的儲存貯體。
   - `manifest_file` - 清單檔案的路徑和檔案名稱

```
#Copyright 2023 Amazon.com, Inc. or its affiliates. All Rights Reserved.
```

```
#PDX-License-Identifier: MIT-0 (For details, see https://github.com/awsdocs/
amazon-rekognition-custom-labels-developer-guide/blob/master/LICENSE-
SAMPLECODE.)


import argparse
import logging
import time
import json
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)

def create_dataset(rek_client, project_arn, dataset_type, bucket,
 manifest_file):
    """
    Creates an Amazon Rekognition Custom Labels dataset.
    :param rek_client: The Amazon Rekognition Custom Labels Boto3 client.
    :param project_arn: The ARN of the project in which you want to create a
 dataset.
    :param dataset_type: The type of the dataset that you want to create (train
 or test).
    :param bucket: The S3 bucket that contains the manifest file.
    :param manifest_file: The path and filename of the manifest file.
    """

    try:
        #Create the project
        logger.info("Creating %s dataset for project %s",dataset_type,
 project_arn)

        dataset_type = dataset_type.upper()

        dataset_source = json.loads(
            '{ "GroundTruthManifest": { "S3Object": { "Bucket": "'
            + bucket
            + '", "Name": "'
            + manifest_file
            + '" } } }'
        )

        response = rek_client.create_dataset(
```

```
            ProjectArn=project_arn, DatasetType=dataset_type,
 DatasetSource=dataset_source
        )

        dataset_arn=response['DatasetArn']

        logger.info("dataset ARN: %s",dataset_arn)

        finished=False
        while finished is False:

            dataset=rek_client.describe_dataset(DatasetArn=dataset_arn)

            status=dataset['DatasetDescription']['Status']

            if status == "CREATE_IN_PROGRESS":
                logger.info("Creating dataset: %s ",dataset_arn)
                time.sleep(5)
                continue

            if status == "CREATE_COMPLETE":
                logger.info("Dataset created: %s", dataset_arn)
                finished=True
                continue

            if status == "CREATE_FAILED":
                error_message = f"Dataset creation failed: {status} :
 {dataset_arn}"
                logger.exception(error_message)
                raise Exception (error_message)

            error_message = f"Failed. Unexpected state for dataset creation:
 {status} : {dataset_arn}"
            logger.exception(error_message)
            raise Exception(error_message)

        return dataset_arn


    except ClientError as err:
        logger.exception("Couldn't create dataset: %s",err.response['Error']
['Message'])
        raise
```

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

    parser.add_argument(
        "project_arn", help="The ARN of the project in which you want to create
 the dataset."
    )

    parser.add_argument(
        "dataset_type", help="The type of the dataset that you want to create
 (train or test)."
    )

    parser.add_argument(
        "bucket", help="The S3 bucket that contains the manifest file."
    )

    parser.add_argument(
        "manifest_file", help="The path and filename of the manifest file."
    )


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()

        print(f"Creating {args.dataset_type} dataset for project
 {args.project_arn}")

        #Create the dataset.
        session = boto3.Session(profile_name='custom-labels-access')
        rekognition_client = session.client("rekognition")

        dataset_arn=create_dataset(rekognition_client,
```

```
            args.project_arn,
            args.dataset_type,
            args.bucket,
            args.manifest_file)

        print(f"Finished creating dataset: {dataset_arn}")


    except ClientError as err:
        logger.exception("Problem creating dataset: %s", err)
        print(f"Problem creating dataset: {err}")



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

Java V2

使用下列值建立資料集。提供下列命令列參數：

- project_arn — 您要為其新增測試資料集之專案的 ARN。
- dataset_type — 您要建立的資料集類型 (train 或 test)。
- bucket - 包含資料集之清單檔案的儲存貯體。
- manifest_file - 清單檔案的路徑和檔案名稱

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

package com.example.rekognition;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.CreateDatasetRequest;
import software.amazon.awssdk.services.rekognition.model.CreateDatasetResponse;
import software.amazon.awssdk.services.rekognition.model.DatasetDescription;
import software.amazon.awssdk.services.rekognition.model.DatasetSource;
```

```
import software.amazon.awssdk.services.rekognition.model.DatasetStatus;
import software.amazon.awssdk.services.rekognition.model.DatasetType;
import software.amazon.awssdk.services.rekognition.model.DescribeDatasetRequest;
import
 software.amazon.awssdk.services.rekognition.model.DescribeDatasetResponse;
import software.amazon.awssdk.services.rekognition.model.GroundTruthManifest;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import software.amazon.awssdk.services.rekognition.model.S3Object;

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

public class CreateDatasetManifestFiles {

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

    public static String createMyDataset(RekognitionClient rekClient, String
 projectArn, String datasetType,
            String bucket, String name) throws Exception, RekognitionException {

        try {

            logger.log(Level.INFO, "Creating {0} dataset for project : {1} from
 s3://{2}/{3} ",
                    new Object[] { datasetType, projectArn, bucket, name });

            DatasetType requestDatasetType = null;

            switch (datasetType) {
            case "train":
                requestDatasetType = DatasetType.TRAIN;
                break;
            case "test":
                requestDatasetType = DatasetType.TEST;
                break;
            default:
                logger.log(Level.SEVERE, "Could not create dataset. Unrecognized
 dataset type: {0}", datasetType);
                throw new Exception("Could not create dataset. Unrecognized
 dataset type: " + datasetType);

            }
```

```
            GroundTruthManifest groundTruthManifest =
GroundTruthManifest.builder()

.s3Object(S3Object.builder().bucket(bucket).name(name).build()).build();

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

            CreateDatasetRequest createDatasetRequest =
CreateDatasetRequest.builder().projectArn(projectArn)

.datasetType(requestDatasetType).datasetSource(datasetSource).build();

            CreateDatasetResponse response =
rekClient.createDataset(createDatasetRequest);

            boolean created = false;

            do {

                DescribeDatasetRequest describeDatasetRequest =
DescribeDatasetRequest.builder()
                        .datasetArn(response.datasetArn()).build();
                DescribeDatasetResponse describeDatasetResponse =
rekClient.describeDataset(describeDatasetRequest);

                DatasetDescription datasetDescription =
describeDatasetResponse.datasetDescription();

                DatasetStatus status = datasetDescription.status();

                logger.log(Level.INFO, "Creating dataset ARN: {0} ",
response.datasetArn());

                switch (status) {

                case CREATE_COMPLETE:
                    logger.log(Level.INFO, "Dataset created");
                    created = true;
                    break;

                case CREATE_IN_PROGRESS:
                    Thread.sleep(5000);
                    break;
```

```
                case CREATE_FAILED:
                    String error = "Dataset creation failed: " +
datasetDescription.statusAsString() + " "
                            + datasetDescription.statusMessage() + " " +
response.datasetArn();
                    logger.log(Level.SEVERE, error);
                    throw new Exception(error);

                default:
                    String unexpectedError = "Unexpected creation state: " +
datasetDescription.statusAsString() + " "
                            + datasetDescription.statusMessage() + " " +
response.datasetArn();
                    logger.log(Level.SEVERE, unexpectedError);
                    throw new Exception(unexpectedError);
                }

            } while (created == false);

            return response.datasetArn();

        } catch (RekognitionException e) {
            logger.log(Level.SEVERE, "Could not create dataset: {0}",
e.getMessage());
            throw e;
        }

    }

    public static void main(String[] args) {

        String datasetType = null;
        String bucket = null;
        String name = null;
        String projectArn = null;
        String datasetArn = null;

        final String USAGE = "\n" + "Usage: " + "<project_arn> <dataset_type>
<dataset_arn>\n\n" + "Where:\n"
                + "   project_arn - the ARN of the project that you want to add
copy the datast to.\n\n"
                + "   dataset_type - the type of the dataset that you want to
create (train or test).\n\n"
```

```
                + "   bucket - the S3 bucket that contains the manifest file.\n
\n"
                + "   name - the location and name of the manifest file within
 the bucket.\n\n";

        if (args.length != 4) {
            System.out.println(USAGE);
            System.exit(1);
        }

        projectArn = args[0];
        datasetType = args[1];
        bucket = args[2];
        name = args[3];

        try {

            // Get the Rekognition client
            RekognitionClient rekClient = RekognitionClient.builder()
                .credentialsProvider(ProfileCredentialsProvider.create("custom-
labels-access"))
                .region(Region.US_WEST_2)
                .build();


             // Create the dataset
            datasetArn = createMyDataset(rekClient, projectArn, datasetType,
 bucket, name);

            System.out.println(String.format("Created dataset: %s",
 datasetArn));

            rekClient.close();

        } catch (RekognitionException rekError) {
            logger.log(Level.SEVERE, "Rekognition client error: {0}",
 rekError.getMessage());
            System.exit(1);
        } catch (Exception rekError) {
            logger.log(Level.SEVERE, "Error: {0}", rekError.getMessage());
            System.exit(1);
        }

    }
```

```
}
```

4. 如果需要新增或變更標籤，請參閱 管理標籤 (SDK)。
5. 請遵循 訓練模型 (SDK) 中的步驟訓練模型。

Amazon SageMaker Ground Truth 工作

有了 Amazon SageMaker Ground Truth，您可以使用 Amazon Mechanical Turk (您選擇的廠商公司) 的員工，或是內部私人員工，以及可讓您建立標記影像集的機器學習。Amazon Rekognition 自訂標籤會從您指定的 Amazon S3 儲存貯體匯入 G SageMaker round Truth 資訊清單檔案。

Amazon Rekognition 自訂標籤支援下列「 SageMaker Ground Truth」任務。

- 影像分類
- 週框方塊

您匯入的檔案是影像和清單檔案。清單檔案包含您匯入影像的標籤和週框方塊資訊。

Amazon Rekognition 需要權限才能存取存放影像的 Amazon S3 儲存貯體。如果您正在使用 Amazon Rekognition 自訂標籤為您設定的主控台儲存貯體，則已設定所需權限。如果您沒有使用主控台儲存貯體，請參閱 存取外部 Amazon S3 儲存貯體。

使用 SageMaker Ground Truth 作業創建清單文件（控制台）

下列程序說明如何使用 SageMaker Ground Truth 工作標記的影像來建立資料集。任務輸出檔案會存放在 Amazon Rekognition 自訂標籤主控台儲存貯體中。

若要使用由「 SageMaker Ground Truth」工作 (主控台) 標記的映像檔建立資料集

1. 登入 AWS Management Console，並開啟位於 https://console.aws.amazon.com/s3/ 的 Amazon S3 主控台。
2. 在主控台儲存貯體中，建立資料夾以保存您的訓練影像。

> **Note**
>
> 您第一次在 AWS 區域中開啟 Amazon Rekognition 自訂標籤主控台時，即會建立該主控台儲存貯體。如需詳細資訊，請參閱 管理 Amazon Rekognition 自訂標籤專案。

3. 將影像上傳至您剛建立的資料夾。
4. 在主控台儲存貯體中，建立資料夾以保存 Ground Truth 任務的輸出。
5. 請在以下位置開啟 SageMaker 主控台。 https://console.aws.amazon.com/sagemaker/
6. 建立 Ground Truth 標記任務。您需要在步驟 2 和步驟 4 中建立之資料夾的 Amazon S3 URL。如需詳細資訊，請參閱使用 Amazon SageMaker Ground Truth 進行資料標籤。
7. 記下 `output.manifest` 檔案在您於步驟 4 建立的資料夾中的位置。它應該位於子資料夾 *`Ground-Truth-Job-Name`*`/manifests/output`。
8. 請遵循 使用 SageMaker Ground Truth 清單文件創建數據集（控制台） 中的指示，使用上傳的資訊清單檔案建立資料集。對於步驟 8，請在 .manifest 檔案位置，輸入您在上一個步驟中記下的位置的 Amazon S3 URL。如果您使用 AWS SDK，請執行 使用 SageMaker Ground Truth 資訊清單檔案 (SDK) 建立資料集。
9. 重複步驟 1-6，為您的測試資料集建立「 SageMaker Ground Truth」工作。

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測試或訓練資料集。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。Amazon Rekognition 自訂標籤支援 SageMaker Ground Truth 資訊清單，並使用下列格式的 JSON 行：

- 分類任務輸出 — 用於將影像層級標籤新增至影像。影像層級標籤會定義影像上的場景、概念或物件類別 (如果不需要物件位置資訊)。一個影像可以有一個以上的影像層級標籤。如需詳細資訊，請參閱清單檔案中的影像層級標籤。
- 週框方塊任務輸出 — 用於在影像上標記一或多個物件的類別和位置。如需詳細資訊，請參閱 清單檔案中的物件本地化。

影像層級和本地化 (週框方塊) JSON Lines 可以鏈接在同一個清單檔案中。

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

匯入清單檔案時，Amazon Rekognition 自訂標籤會套用限制、語法和語意的驗證規則。如需詳細資訊，請參閱 清單檔案的驗證規則。

清單檔案所參考的影像必須位於同一個 Amazon S3 儲存貯體中。清單檔案可以位於與存放影像的 Amazon S3 儲存貯體不同的 Amazon S3 儲存貯體中。您可以在 JSON Line 的 `source-ref` 欄位中指定影像的位置。

Amazon Rekognition 需要權限才能存取存放影像的 Amazon S3 儲存貯體。如果您正在使用 Amazon Rekognition 自訂標籤為您設定的主控台儲存貯體，則已設定所需權限。如果您沒有使用主控台儲存貯體，請參閱 存取外部 Amazon S3 儲存貯體。

主題

- 建立清單檔案
- 清單檔案中的影像層級標籤
- 清單檔案中的物件本地化
- 清單檔案的驗證規則

建立清單檔案

下列程序會建立具有訓練和測試資料集的專案。資料集會從您建立的訓練和測試清單檔案建立。

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

1. 在主控台儲存貯體中，建立資料夾以保存您的清單檔案。
2. 在主控台儲存貯體中，建立資料夾以保存您的影像。
3. 將影像上傳至您剛建立的資料夾。
4. 為您的訓練資料集建立 SageMaker Ground Truth 格式資訊清單檔案。如需詳細資訊，請參閱 清單檔案中的影像層級標籤 及 清單檔案中的物件本地化。

   > **Important**
   >
   > 每個 JSON Line 中的 `source-ref` 欄位值都必須對應至您上傳的影像。

5. 為您的測試資料集建立 SageMaker Ground Truth 格式資訊清單檔案。
6. 將清單檔案上傳至您剛建立的資料夾。
7. 記下清單檔案的位置。

8. 請遵循 使用 SageMaker Ground Truth 清單文件創建數據集（控制台） 中的指示，使用上傳的資訊清單檔案建立資料集。對於步驟 8，請在 .manifest 檔案位置，輸入您在上一個步驟中記下的位置的 Amazon S3 URL。如果您使用 AWS SDK，請執行 使用 SageMaker Ground Truth 資訊清單檔案 (SDK) 建立資料集。

清單檔案中的影像層級標籤

若要匯入影像層級標籤 (標有場景、概念或不需要當地語系化資訊的物件的影像)，您可以將「 SageMaker Ground Truth 分類 Job 輸出」格式化 JSON 行新增至資訊清單檔案。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。

> **Tip**
>
> 為了簡化清單檔案的建立，我們會提供 Python 指令碼，用於從 CSV 檔案建立清單檔案。如需詳細資訊，請參閱 從 CSV 檔案建立清單檔案。。

建立影像層級標籤的清單檔案

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 Line 應該看起來類似下列內容。

```
{"source-ref":"s3://custom-labels-console-us-east-1-nnnnnnnnnn/gt-job/
manifest/IMG_1133.png","TestCLConsoleBucket":0,"TestCLConsoleBucket-
metadata":{"confidence":0.95,"job-name":"labeling-job/
testclconsolebucket","class-name":"Echo Dot","human-annotated":"yes","creation-
date":"2020-04-15T20:17:23.433061","type":"groundtruth/image-classification"}}
```

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

影像層級 JSON Lines

在本區段中，我們會說明如何為單一影像建立 JSON Line。考慮下列影像。下列影像的場景可能稱為 Sunrise。

上一個影像 (具有場景 Sunrise) 的 JSON Line 可能如下所示。

```
{
    "source-ref": "s3://bucket/images/sunrise.png",
    "testdataset-classification_Sunrise": 1,
    "testdataset-classification_Sunrise-metadata": {
        "confidence": 1,
        "job-name": "labeling-job/testdataset-classification_Sunrise",
        "class-name": "Sunrise",
        "human-annotated": "yes",
        "creation-date": "2020-03-06T17:46:39.176",
        "type": "groundtruth/image-classification"
    }
}
```

記下以下資訊。

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。

testdataset-classification_Sunrise

(必要) 屬性標籤。您可以選擇欄位名稱。欄位值 (前面範例中為 1) 為標籤屬性識別碼。Amazon Rekognition 自訂標籤不會使用，而且可以是任何整數值。必須有由欄位名稱識別的對應中繼資料，並附加了 -metadata。例如 "testdataset-classification_Sunrise-metadata"。

testdataset-classification_Sunrise-metadata

(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。

confidence

(必要) Amazon Rekognition 自訂標籤目前未使用，但必須提供介於 0 到 1 之間的值。

job-name

(選用) 您為處理影像的任務選擇的名稱。

class-name

(必要) 您為套用至影像的場景或概念選擇的類別名稱。例如 "Sunrise"。

human-annotated

(必要) 如果註釋由人類完成，則指定 "yes"。否則為 "no"。

creation-date

(必要) 建立標籤時的國際標準時間 (UTC) 日期和時間。

type

(必要) 應套用至影像的處理類型。若為影像層級標籤，值為 "groundtruth/image-classification"。

對影像新增多個影像層級標籤

您可以對影像新增多個標籤。例如，下面的 JSON 對單一影像新增了兩個標籤，即 football 和 ball。

```
{
    "source-ref": "S3 bucket location",
    "sport0":0, # FIRST label
    "sport0-metadata": {
        "class-name": "football",
        "confidence": 0.8,
        "type":"groundtruth/image-classification",
        "job-name": "identify-sport",
        "human-annotated": "yes",
        "creation-date": "2018-10-18T22:18:13.527256"
    },
    "sport1":1, # SECOND label
    "sport1-metadata": {
        "class-name": "ball",
        "confidence": 0.8,
        "type":"groundtruth/image-classification",
        "job-name": "identify-sport",
        "human-annotated": "yes",
        "creation-date": "2018-10-18T22:18:13.527256"
    }
}  # end of annotations for 1 image
```

清單檔案中的物件本地化

您可以將「 SageMakerGround Truth 邊界方框 Job 輸出」格式 JSON 行新增至資訊清單檔案，匯入標有物件當地語系化資訊的影像。

本地化資訊代表物件在影像上的位置。位置由圍繞物件的週框方塊表示。週框方塊結構包含週框方塊的左上角座標，以及週框方塊的寬度和高度。週框方塊格式 JSON Line 包含影像上一或多個物件位置的週框方塊，以及影像上每個物件的類別。

清單檔案由一或多個 JSON Lines 組成，每行包含單一影像的資訊。

建立物件本地化的清單檔案

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 Line 應該看起來類似下列內容。

```
{"source-ref": "s3://bucket/images/IMG_1186.png", "bounding-box": {"image_size":
 [{"width": 640, "height": 480, "depth": 3}], "annotations": [{ "class_id":
 1, "top": 251, "left": 399, "width": 155, "height": 101}, {"class_id": 0,
 "top": 65, "left": 86, "width": 220, "height": 334}]}, "bounding-box-metadata":
```

```
{"objects": [{ "confidence": 1}, {"confidence": 1}], "class-map": {"0": "Echo",
"1": "Echo Dot"}, "type": "groundtruth/object-detection", "human-annotated":
"yes", "creation-date": "2013-11-18T02:53:27", "job-name": "my job"}}
```

3. 儲存檔案。您可以使用副檔名 `.manifest`，但這不是必要的。
4. 使用您剛建立的檔案建立資料集。如需詳細資訊，請參閱 若要使用 SageMaker Ground Truth 格式資訊清單檔案 (主控台) 建立資料集。

物件週框方塊 JSON Lines

在本區段中，我們會說明如何為單一影像建立 JSON Line。下列影像即顯示 Amazon Echo 和 Amazon Echo Dot 裝置周圍的週框方塊。

以下是上一個影像的週框方塊 JSON Line。

```
{
 "source-ref": "s3://custom-labels-bucket/images/IMG_1186.png",
 "bounding-box": {
  "image_size": [{
   "width": 640,
   "height": 480,
   "depth": 3
  }],
  "annotations": [{
   "class_id": 1,
   "top": 251,
   "left": 399,
   "width": 155,
   "height": 101
  }, {
   "class_id": 0,
   "top": 65,
   "left": 86,
   "width": 220,
   "height": 334
  }]
 },
 "bounding-box-metadata": {
  "objects": [{
   "confidence": 1
  }, {
   "confidence": 1
  }],
  "class-map": {
   "0": "Echo",
   "1": "Echo Dot"
  },
  "type": "groundtruth/object-detection",
  "human-annotated": "yes",
  "creation-date": "2013-11-18T02:53:27",
  "job-name": "my job"
 }
}
```

記下以下資訊。

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。

bounding-box

(必要) 屬性標籤。您可以選擇欄位名稱。包含在影像中偵測到的每個物件的影像大小和週框方塊。必須有由欄位名稱識別的對應中繼資料，並附加了 -metadata。例如 "bounding-box-metadata"。

image_size

(必要) 包含以像素為單位之影像大小的單一元素陣列。

- height — (必要) 以像素為單位的影像高度。
- width — (必要) 以像素為單位的影像深度。
- depth — (必要) 影像中的頻道數。若為 RGB 影像，值為 3。Amazon Rekognition 自訂標籤未使用，但需要一個值。

annotations

(必要) 影像中偵測到的每個物件的週框方塊資訊陣列。

- class_id — (必要) 對應至 class-map 中的標籤。在前面的範例中，class_id 為 1 的物件是影像中的 Echo Dot。
- top — (必要) 從影像頂端到週框方塊頂端的距離，以像素為單位。
- left — (必要) 從影像左側到週框方塊左側的距離，以像素為單位。
- width — (必要) 以像素為單位的週框方塊寬度。
- height — (必要) 以像素為單位的週框方塊高度。

bounding-box-metadata

(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。影像中偵測到的每個物件的週框方塊資訊陣列。

物件

(必要) 影像中的物件陣列。對應至註釋陣列 (依索引)。Amazon Rekognition 自訂標籤不使用可信度屬性。

class-map

(必要) 套用至影像中偵測到之物件的類別的對應。

type

(必要) 分類任務的類型。"groundtruth/object-detection" 將任務識別為物件偵測。

creation-date

(必要) 建立標籤時的國際標準時間 (UTC) 日期和時間。

human-annotated

(必要) 如果註釋由人類完成，則指定 "yes"。否則為 "no"。

job-name

(必要) 處理影像的任務的名稱。

清單檔案的驗證規則

匯入清單檔案時，Amazon Rekognition 自訂標籤會套用限制、語法和語意的驗證規則。 SageMaker Ground Truth 模式強制執行語法驗證。如需詳細資訊，請參閱輸出。以下是限制和語意的驗證規則。

> **Note**
>
> - 20% 的無效規則會累加套用至所有驗證規則。如果由於任何組合而導致匯入超過 20% 的限制，例如 15% 無效 JSON 和 15% 無效影像，則匯入會失敗。
> - 每個資料集物件都是清單檔案中的一行。空白/無效行也會計為資料集物件。
> - 重疊是 (測試和訓練之間的常見標籤)/(訓練標籤)。

主題

- 限制
- 語意學

限制

驗證	限制	引發錯誤
清單檔案大小	上限為 1 GB	錯誤
清單檔案的最大行計數	清單檔案中最多 250,000 個資料集物件的行。	錯誤
每個標籤的有效資料集物件總數的下邊界	>= 1	錯誤
標籤上的下邊界	>=2	錯誤
標籤上的上邊界	<= 250	錯誤
每個影像的最小週框方塊	0	無
每個影像的最大週框方塊	50	無

語意學

驗證	限制	引發錯誤
空白清單檔案		錯誤
缺少/無法存取的 source-ref 物件	物件數量小於 20%	警告
缺少/無法存取的 source-ref 物件	物件數量 > 20%	錯誤
訓練資料集中不存在測試標籤	標籤中至少有 50% 的重疊	錯誤
資料集中相同標籤的標籤與物件範例的混合。資料集物件中相同類別的分類和偵測。		沒有錯誤或警告

驗證	限制	引發錯誤
測試和訓練之間的重疊資產	測試和訓練資料集之間不應該有重疊。	
資料集中的影像必須來自同一個儲存貯體	如果物件位於不同的儲存貯體中，則會發生錯誤	錯誤

將其他資料集格式轉換為清單檔案

您可以使用下列資訊，從各種來源資料集 SageMaker 格式建立 Amazon 格式資訊清單檔案。建立清單檔案後，請使用它來建立資料集。如需詳細資訊，請參閱 清單檔案。

主題

- 轉換 COCO 資料集
- 轉換多標籤 SageMaker Ground Truth 清單文件
- 從 CSV 檔案建立清單檔案。

轉換 COCO 資料集

COCO 是一種用於指定大規模物件偵測、分割和字幕資料集的格式。此 Python 範例會說明如何將 COCO 物件偵測格式資料集轉換為 Amazon Rekognition 自訂標籤週框方塊格式清單檔案。本區段還包括您可用於撰寫自己的程式碼的資訊。

COCO 格式的 JSON 檔案由五個區段組成，提供整個資料集的資訊。如需詳細資訊，請參閱 COCO 格式。

- `info` — 有關資料集的一般資訊。
- `licenses` — 資料集中影像的授權資訊。
- `images` — 資料集中的影像清單。
- `annotations` — 資料集中所有影像中出現的註解清單 (包括週框方塊)。
- `categories` — 標籤類別清單。

您需要 `images`、`annotations` 和 `categories` 清單中的資訊，才能建立 Amazon Rekognition 自訂標籤清單檔案。

Amazon Rekognition 自訂標籤清單檔案採用 JSON Lines 格式，其中每行都具有影像上一或多個物件的週框方塊和標籤資訊。如需詳細資訊，請參閱 清單檔案中的物件本地化。

將 COCO 物件對應至自訂標籤 JSON Line

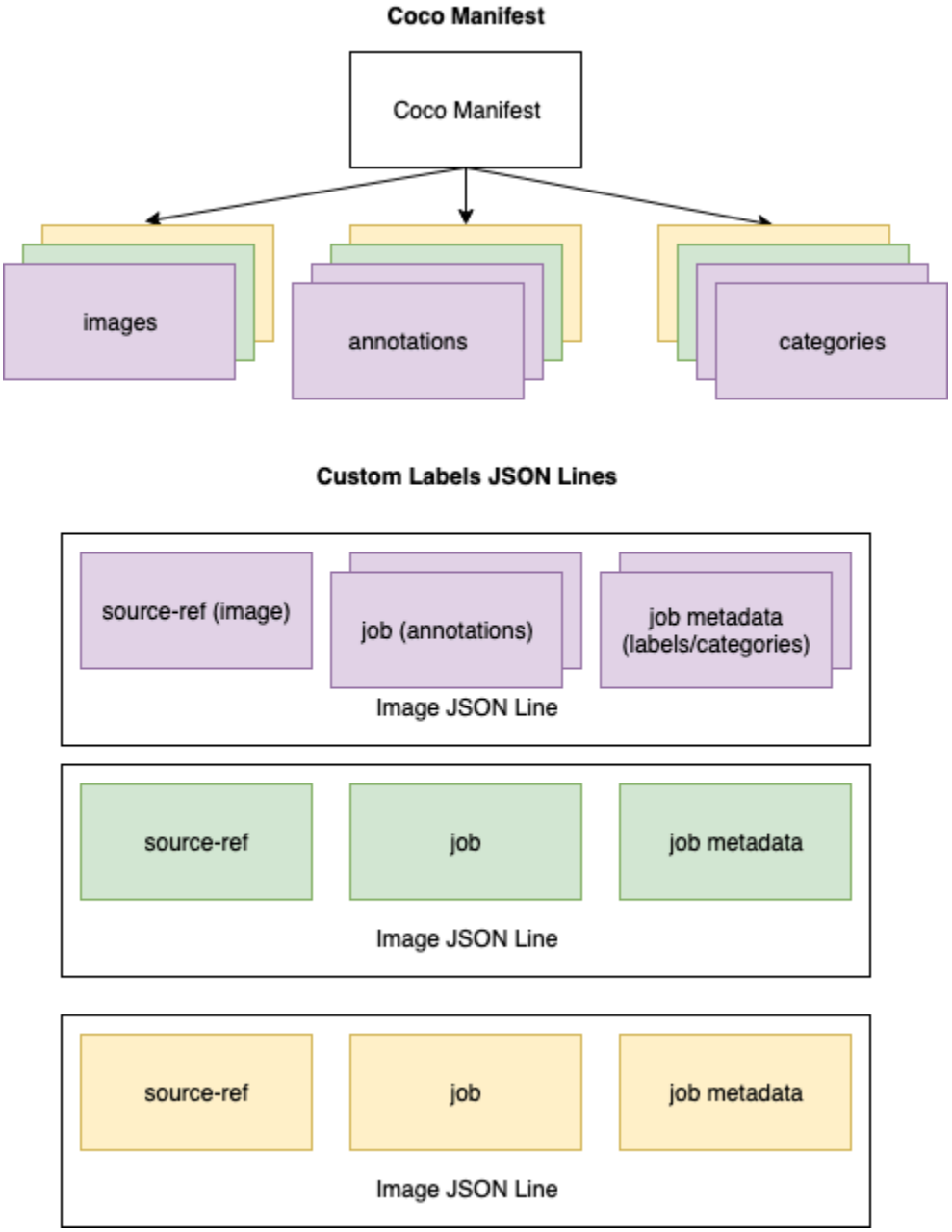
若要轉換 COCO 格式資料集，請將 COCO 資料集對應至 Amazon Rekognition 自訂標籤清單檔案，以進行物件本地化。如需詳細資訊，請參閱 清單檔案中的物件本地化。若要為每個影像建置 JSON Line，清單檔案需要對應 COCO 資料集 `image`、`annotation` 和 `category` 物件欄位 ID。

以下是 COCO 清單檔案範例。如需詳細資訊，請參閱 COCO 格式。

```
{
    "info": {
        "description": "COCO 2017 Dataset","url": "http://cocodataset.org","version":
 "1.0","year": 2017,"contributor": "COCO Consortium","date_created": "2017/09/01"
    },
    "licenses": [
        {"url": "http://creativecommons.org/licenses/by/2.0/","id": 4,"name":
 "Attribution License"}
    ],
    "images": [
        {"id": 242287, "license": 4, "coco_url": "http://images.cocodataset.org/
val2017/xxxxxxxxxxxx.jpg", "flickr_url": "http://farm3.staticflickr.com/2626/
xxxxxxxxxxxx.jpg", "width": 426, "height": 640, "file_name": "xxxxxxxxx.jpg",
 "date_captured": "2013-11-15 02:41:42"},
        {"id": 245915, "license": 4, "coco_url": "http://images.cocodataset.org/
val2017/nnnnnnnnnnnn.jpg", "flickr_url": "http://farm1.staticflickr.com/88/
xxxxxxxxxxxx.jpg", "width": 640, "height": 480, "file_name": "nnnnnnnnnn.jpg",
 "date_captured": "2013-11-18 02:53:27"}
    ],
    "annotations": [
        {"id": 125686, "category_id": 0, "iscrowd": 0, "segmentation": [[164.81,
 417.51,.....167.55, 410.64]], "image_id": 242287, "area": 42061.80340000001, "bbox":
 [19.23, 383.18, 314.5, 244.46]},
        {"id": 1409619, "category_id": 0, "iscrowd": 0, "segmentation": [[376.81,
 238.8,........382.74, 241.17]], "image_id": 245915, "area": 3556.2197000000015,
 "bbox": [399, 251, 155, 101]},
        {"id": 1410165, "category_id": 1, "iscrowd": 0, "segmentation": [[486.34,
 239.01,..........495.95, 244.39]], "image_id": 245915, "area": 1775.8932499999994,
 "bbox": [86, 65, 220, 334]}
    ],
    "categories": [
        {"supercategory": "speaker","id": 0,"name": "echo"},
```

```
        {"supercategory": "speaker","id": 1,"name": "echo dot"}
    ]
}
```

下圖顯示資料集的 COCO 資料集清單如何對應至影像的 Amazon Rekognition 自訂標籤 JSON Lines。相符顏色表示單一影像的資訊。



取得單一 JSON Line 的 COCO 物件

1. 對於影像清單中的每個影像，請從註釋清單中取得註釋，其中註釋欄位 `image_id` 的值會和影像 `id` 欄位相符。
2. 對於步驟 1 中相符的每個註釋，請詳閱 `categories` 清單並取得 `category` 欄位 `id` 的值和 `annotation` 物件 `category_id` 欄位相符的每個 `category`。
3. 使用相符的 `image`、`annotation` 和 `category` 物件為影像建立 JSON Line。若要對應欄位，請參閱 將 COCO 物件對應至自訂標籤 JSON Line 物件欄位。
4. 重複步驟 1 至 3，直到您為 `images` 清單中的每個 `image` 物件建立 JSON Lines 為止。

如需範例程式碼，請參閱 轉換 COCO 資料集。

將 COCO 物件對應至自訂標籤 JSON Line 物件欄位

在您識別 Amazon Rekognition 自訂標籤 JSON Line 的 COCO 物件之後，您需要將 COCO 物件欄位對應至各自的 Amazon Rekognition 自訂標籤 JSON Line 物件欄位。下列 Amazon Rekognition 自訂標籤 JSON Line 範例會將一個影像 (`id=000000245915`) 對應至前一個 COCO JSON 範例。記下以下資訊。

- `source-ref` 是 Amazon S3 儲存貯體中影像的位置。如果您的 COCO 影像未存放在 Amazon S3 儲存貯體中，您需要將它們移至 Amazon S3 儲存貯體。
- 此 `annotations` 清單包含影像上每個物件的 `annotation` 物件。`annotation` 物件包括週框方塊資訊 (`top`、`left`、`width`、`height`) 和標籤識別碼 (`class_id`)。
- 標籤識別碼 (`class_id`) 會對應至中繼資料中的 `class-map` 清單。它會列出在影像上使用的標籤。

```
{
 "source-ref": "s3://custom-labels-bucket/images/000000245915.jpg",
 "bounding-box": {
  "image_size": {
   "width": 640,
   "height": 480,
   "depth": 3
  },
  "annotations": [{
   "class_id": 0,
   "top": 251,
   "left": 399,
   "width": 155,
```

```
    "height": 101
   }, {
    "class_id": 1,
    "top": 65,
    "left": 86,
    "width": 220,
    "height": 334
   }]
  },
  "bounding-box-metadata": {
   "objects": [{
    "confidence": 1
   }, {
    "confidence": 1
   }],
   "class-map": {
    "0": "Echo",
    "1": "Echo Dot"
   },
   "type": "groundtruth/object-detection",
   "human-annotated": "yes",
   "creation-date": "2018-10-18T22:18:13.527256",
   "job-name": "my job"
  }
}
```

使用下列資訊將 Amazon Rekognition 自訂標籤清單檔案欄位對應至 COCO 資料集 JSON 欄位。

source-ref

影像位置的 S3 格式 URL。影片必須存放在 S3 儲存貯體中。如需詳細資訊，請參閱 source-ref。如果 coco_url COCO 欄位指向 S3 儲存貯體位置，您可以使用 coco_url 的值作為 source-ref 的值。或者，您可以將 source-ref 對應至 file_name (COCO) 欄位，並在轉換程式碼中，將必要的 S3 路徑新增至影像的存放位置。

bounding-box

您選擇的標籤屬性名稱。如需詳細資訊，請參閱 *bounding-box*。

image_size

影像大小 (以像素為單位)。對應至image影像清單中的 物件。

- height-> image.height

- width-> image.width
- depth-> Amazon Rekognition 自訂標籤目前未使用，但必須一個值。

註釋

annotation 物件的清單。影像上的每個物件都有一個 annotation。

註釋

包含影像上某個物件執行個體的週框方塊資訊。

- class_id -> 數字 ID 對應至自訂標籤的 class-map 清單。
- top -> bbox[1]
- left -> bbox[0]
- width -> bbox[2]
- height -> bbox[3]

bounding-box-metadata

標籤屬性的中繼資料。包括標籤和標籤識別碼。如需詳細資訊，請參閱 *bounding-box*-metadata。

物件

影像中的物件陣列。對應至 annotations 清單 (依索引)。

物件

- confidence->Amazon Rekognition 自訂標籤未使用，但需要值 (1)。

class-map

套用至影像中偵測到之物件的標籤 (類別) 的對應。對應至類別清單中的類別物件。

- id -> category.id
- id value -> category.name

type

必須為 groundtruth/object-detection

human-annotated

可指定為 yes 或 no。如需詳細資訊，請參閱 *bounding-box*-metadata。

creation-date -> image.date_captured

影像的建立日期和時間。對應至 COCO 影像清單中影像的 image.date_captured 欄位。Amazon Rekognition 自訂標籤預期 creation-date 的格式為 Y-M-DTH:M:S。

job-name

您選擇的任務名稱。

COCO 格式

COCO 資料集由五個部分的資訊組成，可提供整個資料集的資訊。COCO 物件偵測資料集的格式會以 COCO 資料格式記錄。

- info — 有關資料集的一般資訊。
- license — 資料集中影像的授權資訊。
- images — 資料集中的影像清單。
- annotations — 資料集中所有影像中出現的註釋清單 (包括週框方塊)。
- categories — 標籤類別清單。

若要建立自訂標籤清單檔案，請使用 COCO 清單檔案中的 images、annotations、和 categories 清單。其他區段 (info、licences) 則非必要。以下是 COCO 清單檔案範例。

```
{
    "info": {
        "description": "COCO 2017 Dataset","url": "http://cocodataset.org","version":
 "1.0","year": 2017,"contributor": "COCO Consortium","date_created": "2017/09/01"
    },
    "licenses": [
        {"url": "http://creativecommons.org/licenses/by/2.0/","id": 4,"name":
 "Attribution License"}
    ],
    "images": [
        {"id": 242287, "license": 4, "coco_url": "http://images.cocodataset.org/
val2017/xxxxxxxxxxxx.jpg", "flickr_url": "http://farm3.staticflickr.com/2626/
```

```
xxxxxxxxxxxx.jpg", "width": 426, "height": 640, "file_name": "xxxxxxxxx.jpg",
 "date_captured": "2013-11-15 02:41:42"},
        {"id": 245915, "license": 4, "coco_url": "http://images.cocodataset.org/
val2017/nnnnnnnnnnnn.jpg", "flickr_url": "http://farm1.staticflickr.com/88/
xxxxxxxxxxx.jpg", "width": 640, "height": 480, "file_name": "nnnnnnnnnn.jpg",
 "date_captured": "2013-11-18 02:53:27"}
    ],
    "annotations": [
        {"id": 125686, "category_id": 0, "iscrowd": 0, "segmentation": [[164.81,
 417.51,......167.55, 410.64]], "image_id": 242287, "area": 42061.80340000001, "bbox":
 [19.23, 383.18, 314.5, 244.46]},
        {"id": 1409619, "category_id": 0, "iscrowd": 0, "segmentation": [[376.81,
 238.8,........382.74, 241.17]], "image_id": 245915, "area": 3556.2197000000015,
 "bbox": [399, 251, 155, 101]},
        {"id": 1410165, "category_id": 1, "iscrowd": 0, "segmentation": [[486.34,
 239.01,..........495.95, 244.39]], "image_id": 245915, "area": 1775.8932499999994,
 "bbox": [86, 65, 220, 334]}
    ],
    "categories": [
        {"supercategory": "speaker","id": 0,"name": "echo"},
        {"supercategory": "speaker","id": 1,"name": "echo dot"}
    ]
}
```

影像清單

COCO 資料集所參考的影像會列在影像陣列中。每個影像物件都包含影像的相關資訊，例如影像檔案名稱。在下列影像物件範例中，請注意下列資訊，以及建立 Amazon Rekognition 自訂標籤清單檔案所需的欄位。

- `id` — (必要) 影像的唯一識別碼。`id` 欄位會對應至註解陣列中的 `id` 欄位 (存放週框方塊資訊的位置)。
- `license` — (非必要) 對應至授權陣列。
- `coco_url` — (選用) 影像的位置
- `flickr_url` — (非必要) 影像在 Flickr 上的位置。
- `width` — (必要) 影像的寬度。
- `height` — (必要) 影像的寬度。
- `file_name` — (必要) 影像檔案名稱。在這個範例中，`file_name` 和 `id` 相符，但這並非 COCO 資料集的需求。
- `date_captured` — (必要) 擷取影像的日期和時間。

```
{
    "id": 245915,
    "license": 4,
    "coco_url": "http://images.cocodataset.org/val2017/nnnnnnnnnnnn.jpg",
    "flickr_url": "http://farm1.staticflickr.com/88/nnnnnnnnnnnnnnnnnn.jpg",
    "width": 640,
    "height": 480,
    "file_name": "000000245915.jpg",
    "date_captured": "2013-11-18 02:53:27"
}
```

註釋 (週框方塊) 清單

所有影像上所有物件的週框方塊資訊會存放在註解清單中。單一註釋物件包含單一物件的週框方塊資訊，以及影像上物件的標籤。影像上物件的每個執行個體都有註釋物件。

在下列範例中，請注意下列資訊，以及建立 Amazon Rekognition 自訂標籤清單檔案所需的欄位。

- id — (非必要) 註釋的識別碼。
- image_id — (必要) 對應於影像陣列中的影像 id。
- category_id — (必要) 標籤的識別碼，可識別週框方塊內的物件。它會對應至類別陣列的 id 欄位。
- iscrowd — (非必要) 指定影像是否包含一群物件。
- segmentation — (非必要) 影像上物件的分割資訊。Amazon Rekognition 自訂標籤不支援分割。
- area — (非必要) 註釋的區域。
- bbox— (必要) 包含影像上物件周圍週框方塊的座標 (以像素為單位)。

```
{
    "id": 1409619,
    "category_id": 1,
    "iscrowd": 0,
    "segmentation": [
        [86.0, 238.8,.........382.74, 241.17]
    ],
    "image_id": 245915,
    "area": 3556.2197000000015,
    "bbox": [86, 65, 220, 334]
}
```

類別清單

標籤資訊存放在類別陣列中。在下列類別物件範例中，請注意下列資訊，以及建立 Amazon Rekognition 自訂標籤清單檔案所需的欄位。

- `supercategory` — (非必要) 標籤的父類別。
- `id` — (必要) 標籤識別碼。`id` 欄位會對應至 `annotation` 物件中的 `category_id` 欄位。在下列範例中，Echo Dot 的識別碼為 2。
- `name` — (必要) 標籤名稱。

```
        {"supercategory": "speaker","id": 2,"name": "echo dot"}
```

轉換 COCO 資料集

使用下列 Python 範例將週框方塊資訊從 COCO 格式資料集轉換為 Amazon Rekognition 自訂標籤清單檔案。程式碼會將建立的清單檔案上傳至 Amazon S3 儲存貯體。此程式碼也會提供 AWS CLI 命令，您可以用來上傳影像。

轉換 COCO 資料集 (SDK)

1. 如果您尚未執行：

   a. 請確認您具備 `AmazonS3FullAccess` 權限。如需詳細資訊，請參閱 設定 SDK 權限。

   b. 安裝並配置 AWS CLI 和 AWS SDKs。如需詳細資訊，請參閱 步驟 4：設定 AWS CLI 和 AWS SDKs。

2. 使用下列 Python 程式碼來轉換 COCO 資料集。設定下列值。

   - `s3_bucket` — 您要存放影像和 Amazon Rekognition 自訂標籤清單檔案之 S3 儲存貯體的名稱。
   - `s3_key_path_images` — 要在 S3 儲存貯體 (`s3_bucket`) 中放置影像的路徑。
   - `s3_key_path_manifest_file` — 要在 S3 儲存貯體 (`s3_bucket`) 中放置自訂標籤清單檔案的路徑。
   - `local_path` — 範例開啟輸入 COCO 資料集的本機路徑，並儲存新的自訂標籤清單檔案。
   - `local_images_path` — 要用於訓練之影像的本機路徑。
   - `coco_manifest` — 輸入 COCO 資料集檔案名稱。

- cl_manifest_file — 範例所建立之清單檔案的名稱。檔案會儲存在 local_path 所指定的位置。按照慣例，該檔案會具有副檔名 .manifest，但這不是必要的。
- job_name — 自訂標籤任務的名稱。

```
import json
import os
import random
import shutil
import datetime
import botocore
import boto3
import PIL.Image as Image
import io

#S3 location for images
s3_bucket = 'bucket'
s3_key_path_manifest_file = 'path to custom labels manifest file/'
s3_key_path_images = 'path to images/'
s3_path='s3://' + s3_bucket  + '/' + s3_key_path_images
s3 = boto3.resource('s3')

#Local file information
local_path='path to input COCO dataset and output Custom Labels manifest/'
local_images_path='path to COCO images/'
coco_manifest = 'COCO dataset JSON file name'
coco_json_file = local_path + coco_manifest
job_name='Custom Labels job name'
cl_manifest_file = 'custom_labels.manifest'

label_attribute ='bounding-box'

open(local_path + cl_manifest_file, 'w').close()

# class representing a Custom Label JSON line for an image
class cl_json_line:
    def __init__(self,job, img):

        #Get image info. Annotations are dealt with seperately
        sizes=[]
        image_size={}
        image_size["width"] = img["width"]
        image_size["depth"] = 3
```

```
        image_size["height"] = img["height"]
        sizes.append(image_size)

        bounding_box={}
        bounding_box["annotations"] = []
        bounding_box["image_size"] = sizes

        self.__dict__["source-ref"] = s3_path + img['file_name']
        self.__dict__[job] = bounding_box

        #get metadata
        metadata = {}
        metadata['job-name'] = job_name
        metadata['class-map'] = {}
        metadata['human-annotated']='yes'
        metadata['objects'] = []
        date_time_obj = datetime.datetime.strptime(img['date_captured'], '%Y-%m-%d
 %H:%M:%S')
        metadata['creation-date']= date_time_obj.strftime('%Y-%m-%dT%H:%M:%S')
        metadata['type']='groundtruth/object-detection'

        self.__dict__[job + '-metadata'] = metadata


print("Getting image, annotations, and categories from COCO file...")

with open(coco_json_file) as f:

    #Get custom label compatible info
    js = json.load(f)
    images = js['images']
    categories = js['categories']
    annotations = js['annotations']

    print('Images: ' + str(len(images)))
    print('annotations: ' + str(len(annotations)))
    print('categories: ' + str(len (categories)))


print("Creating CL JSON lines...")

images_dict = {image['id']: cl_json_line(label_attribute, image) for image in
 images}
```

```
print('Parsing annotations...')
for annotation in annotations:

    image=images_dict[annotation['image_id']]

    cl_annotation = {}
    cl_class_map={}

    # get bounding box information
    cl_bounding_box={}
    cl_bounding_box['left'] = annotation['bbox'][0]
    cl_bounding_box['top'] = annotation['bbox'][1]

    cl_bounding_box['width'] = annotation['bbox'][2]
    cl_bounding_box['height'] = annotation['bbox'][3]
    cl_bounding_box['class_id'] = annotation['category_id']

    getattr(image, label_attribute)['annotations'].append(cl_bounding_box)


    for category in categories:
         if annotation['category_id'] == category['id']:
            getattr(image, label_attribute + '-metadata')['class-map']
[category['id']]=category['name']


    cl_object={}
    cl_object['confidence'] = int(1)  #not currently used by Custom Labels
    getattr(image, label_attribute + '-metadata')['objects'].append(cl_object)

print('Done parsing annotations')

# Create manifest file.
print('Writing Custom Labels manifest...')

for im in images_dict.values():

    with open(local_path+cl_manifest_file, 'a+') as outfile:
            json.dump(im.__dict__,outfile)
            outfile.write('\n')
            outfile.close()

# Upload manifest file to S3 bucket.
print ('Uploading Custom Labels manifest file to S3 bucket')
```

```
print('Uploading'  + local_path + cl_manifest_file + ' to ' +
 s3_key_path_manifest_file)
print(s3_bucket)
s3 = boto3.resource('s3')
s3.Bucket(s3_bucket).upload_file(local_path + cl_manifest_file,
 s3_key_path_manifest_file + cl_manifest_file)

# Print S3 URL to manifest file,
print ('S3 URL Path to manifest file. ')
print('\033[1m s3://' + s3_bucket + '/' + s3_key_path_manifest_file +
 cl_manifest_file + '\033[0m')

# Display aws s3 sync command.
print ('\nAWS CLI s3 sync command to upload your images to S3 bucket. ')
print ('\033[1m aws s3 sync ' + local_images_path + ' ' + s3_path + '\033[0m')
```

3. 執行程式碼。
4. 在程式輸出中，記下磁碟區 `s3 sync` 命令。下一個步驟需要此值。
5. 在命令提示中，執行 `s3 sync` 命令。將影像上傳至 S3 儲存貯體。如果命令在上傳期間失敗，請再次執行，直到本機影像與 S3 儲存貯體同步為止。
6. 在程式輸出中，記下清單檔案的 S3 URL 路徑。下一個步驟需要此值。
7. 請遵循 使用 SageMaker Ground Truth 清單文件創建數據集（控制台） 中的指示，使用上傳的清單檔案建立資料集。對於步驟 8，請在 .manifest 檔案位置，輸入您在上一個步驟中記下的 Amazon S3 URL。如果您使用 AWS SDK，請執行 使用 SageMaker Ground Truth 資訊清單檔案 (SDK) 建立資料集。

轉換多標籤 SageMaker Ground Truth 清單文件

本主題說明如何將多標籤 Amazon SageMaker Ground Truth 資訊清單檔案轉換為 Amazon Rekognition 自訂標籤格式資訊清單檔案。

SageMaker 多標籤任務的 Ground Truth 資訊清單檔案的格式與 Amazon Rekognition 自訂標籤格式資訊清單檔案的格式不同。多標籤分類指將影像分類為一組類別，但可能同時屬於多個類別。在這種情況下，影像可能有多個標籤 (多標籤)，例如 football 和 ball。

如需多標籤「 SageMaker Ground Truth」工作的相關資訊，請參閱影像分類 (多標籤)。如需多標籤格式 Amazon Rekognition 自訂標籤清單檔案的相關資訊，請參閱 the section called “對影像新增多個影像層級標籤”。

獲取 SageMaker Ground Truth 工作的清單文件

下列程序說明如何取得 Amazon SageMaker Ground Truth 工作的輸出資訊清單檔案 (`output.manifest`)。將 `output.manifest` 用為下個程序的輸入。

下載 SageMaker Ground Truth 工作資訊清單檔案

1. 開啟 https://console.aws.amazon.com/sagemaker/。
2. 在導覽窗格中，選擇 Ground Truth，然後選擇標記任務。
3. 選擇包含您要使用之清單檔案的標記任務。
4. 在詳細資訊頁面上，選擇輸出資料集位置下方的連結。Amazon S3 主控台會在資料集位置開啟。
5. 選擇 `Manifests`、`output`，然後選擇 `output.manifest`。
6. 若要下載清單檔案，請選擇物件動作，然後選擇下載。

轉換多標籤 SageMaker 資訊清單檔案

下列程序會從現有的多標籤格式資訊清單檔案建立多標籤格式的 Amazon Rekognition 自訂標籤資訊清單檔案。 SageMaker GroundTruth

> **Note**
>
> 若要執行程式碼，您需要 Python 版本 3 或更高版本。

若要轉換多標籤 SageMaker 資訊清單檔案

1. 使用以下 Python 程式碼。提供您在 獲取 SageMaker Ground Truth 工作的清單文件 中建立的清單檔案名稱作為命令列引數。

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier:  Apache-2.0
"""
Purpose
Shows how to create and Amazon Rekognition Custom Labels format
manifest file from an Amazon SageMaker Ground Truth Image
Classification (Multi-label) format manifest file.
"""
import json
import logging
```

```
import argparse
import os.path

logger = logging.getLogger(__name__)

def create_manifest_file(ground_truth_manifest_file):
    """
    Creates an Amazon Rekognition Custom Labels format manifest file from
    an Amazon SageMaker Ground Truth Image Classification (Multi-label) format
    manifest file.
    :param: ground_truth_manifest_file: The name of the Ground Truth manifest file,
    including the relative path.
    :return: The name of the new Custom Labels manifest file.
    """

    logger.info('Creating manifest file from %s', ground_truth_manifest_file)
    new_manifest_file =
 f'custom_labels_{os.path.basename(ground_truth_manifest_file)}'

    # Read the SageMaker Ground Truth manifest file into memory.
    with open(ground_truth_manifest_file) as gt_file:
        lines = gt_file.readlines()

    #Iterate through the lines one at a time to generate the
    #new lines for the Custom Labels manifest file.
    with open(new_manifest_file, 'w') as the_new_file:
        for line in lines:
            #job_name - The of the Amazon Sagemaker Ground Truth job.
            job_name = ''
            # Load in the old json item from the Ground Truth manifest file
            old_json = json.loads(line)

            # Get the job name
            keys = old_json.keys()
            for key in keys:
                if 'source-ref' not in key and '-metadata' not in key:
                    job_name = key

            new_json = {}
            # Set the location of the image
            new_json['source-ref'] = old_json['source-ref']

            # Temporarily store the list of labels
            labels = old_json[job_name]
```

```
            # Iterate through the labels and reformat to Custom Labels format
            for index, label in enumerate(labels):
                new_json[f'{job_name}{index}'] = index
                metadata = {}
                metadata['class-name'] = old_json[f'{job_name}-metadata']['class-
map'][str(label)]
                metadata['confidence'] = old_json[f'{job_name}-metadata']
['confidence-map'][str(label)]
                metadata['type'] = 'groundtruth/image-classification'
                metadata['job-name'] = old_json[f'{job_name}-metadata']['job-name']
                metadata['human-annotated'] = old_json[f'{job_name}-metadata']
['human-annotated']
                metadata['creation-date'] = old_json[f'{job_name}-metadata']
['creation-date']
                # Add the metadata to new json line
                new_json[f'{job_name}{index}-metadata'] = metadata
            # Write the current line to the json file
            the_new_file.write(json.dumps(new_json))
            the_new_file.write('\n')

    logger.info('Created %s', new_manifest_file)
    return  new_manifest_file

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

    parser.add_argument(
        "manifest_file", help="The Amazon SageMaker Ground Truth manifest file"
        "that you want to use."
    )


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()
```

```
        # Create the manifest file
        manifest_file = create_manifest_file(args.manifest_file)
        print(f'Manifest file created: {manifest_file}')
    except FileNotFoundError as err:
        logger.exception('File not found: %s', err)
        print(f'File not found: {err}. Check your manifest file.')

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

2. 記下指令碼顯示的新清單檔案的名稱。在下一個步驟中用得到。
3. 將清單檔案上傳到您要用於存放清單檔案的 Amazon S3 儲存貯體。

> **Note**
>
> 請確保 Amazon Rekognition 自訂標籤可存取清單檔案 JSON Lines 的 `source-ref` 欄位中參考的 Amazon S3 儲存貯體。如需詳細資訊，請參閱 存取外部 Amazon S3 儲存貯體。如果您的 Ground Truth 任務將影像存放在 Amazon Rekognition 自訂標籤主控台儲存貯體中，則不需要新增權限。

4. 請遵循 使用 SageMaker Ground Truth 清單文件創建數據集（控制台） 中的指示，使用上傳的資訊清單檔案建立資料集。對於步驟 8，請在 .manifest 檔案位置，輸入清單檔案位置的 Amazon S3 URL。如果您使用 AWS SDK，請執行 使用 SageMaker Ground Truth 資訊清單檔案 (SDK) 建立資料集。

從 CSV 檔案建立清單檔案。

此 Python 指令碼範例使用逗號分隔值 (CSV) 檔案來標記影像，簡化了清單檔案的建立。建立 CSV 檔案。清單檔案適用於多標籤影像分類 或 多標籤圖像分類。如需詳細資訊，請參閱 尋找物件、場景和概念。

> **Note**
>
> 此指令碼不會建立適合尋找物件位置或尋找品牌位置的清單檔案。

清單檔案會描述用於訓練模型的影像。例如，指派給影像的影像位置和標籤。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 the section called “清單檔案中的影像層級標籤”。

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 檔案中的每一資料列會代表單一影像，並對應至清單檔案中的 JSON Line。若要為支援多標籤影像分類的清單檔案建立 CSV 檔案，請在每一資料列新增一或多個影像層級標籤。若要建立適合 影像分類 的清單檔案，請將單一影像層級標籤新增至每一資料列。

例如，下列 CSV 檔案描述 多標籤圖像分類 (Flowers) 入門專案中的影像。

```
camellia1.jpg,camellia,with_leaves
camellia2.jpg,camellia,with_leaves
camellia3.jpg,camellia,without_leaves
helleborus1.jpg,helleborus,without_leaves,not_fully_grown
helleborus2.jpg,helleborus,with_leaves,fully_grown
helleborus3.jpg,helleborus,with_leaves,fully_grown
jonquil1.jpg,jonquil,with_leaves
jonquil2.jpg,jonquil,with_leaves
jonquil3.jpg,jonquil,with_leaves
jonquil4.jpg,jonquil,without_leaves
mauve_honey_myrtle1.jpg,mauve_honey_myrtle,without_leaves
mauve_honey_myrtle2.jpg,mauve_honey_myrtle,with_leaves
mauve_honey_myrtle3.jpg,mauve_honey_myrtle,with_leaves
mediterranean_spurge1.jpg,mediterranean_spurge,with_leaves
mediterranean_spurge2.jpg,mediterranean_spurge,without_leaves
```

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 (`camellia1.jpg,camellia,with_leaves`) 的 JSON Line。

```
{"source-ref": "s3://bucket/flowers/train/camellia1.jpg","camellia": 1,"camellia-
metadata":{"confidence": 1,"job-name": "labeling-job/camellia","class-name":
 "camellia","human-annotated": "yes","creation-date": "2022-01-21T14:21:05","type":
 "groundtruth/image-classification"},"with_leaves": 1,"with_leaves-metadata":
{"confidence": 1,"job-name": "labeling-job/with_leaves","class-name":
 "with_leaves","human-annotated": "yes","creation-date": "2022-01-21T14:21:05","type":
 "groundtruth/image-classification"}}
```

在 CSV 範例中，影像的 Amazon S3 路徑不存在。如果您的 CSV 檔案不包含影像的 Amazon S3 路徑，請使用 `--s3_path` 命令列引數指定影像的 Amazon S3 路徑。

指令碼會在已刪除重複影像的 CSV 檔案中記錄每個影像的第一個項目。已刪除重複影像的 CSV 檔案包含輸入 CSV 檔案中找到的每個影像的單一執行個體。輸入 CSV 檔案中影像的進一步出現次數會記錄在重複影像 CSV 檔案中。如果指令碼尋找重複影像，請檢閱重複影像的 CSV 檔案，並視需要更新已刪除重複影像的 CSV 檔案。使用已刪除重複資料的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，則指令碼會刪除已刪除重複影像的 CSV 檔案和重複影像的 CSV 檔案，因為它們是空白的。

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。

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	欄位 n
影像名稱或 Amazon S3 路徑影像。例如 `s3://my-bucket/flowers/train/camellia1.jpg` 。您不能混合使用具有 Amazon S3 路徑的影像和不具有 Amazon S3 路徑的影像。	影像的第一個影像層級標籤。	一或多個以逗號分隔的其他影像層級標籤。只有在您想要建立支援多標籤影像分類的清單檔案時才新增。

   例如：`camellia1.jpg,camellia,with_leaves` 或 `s3://my-bucket/flowers/train/camellia1.jpg,camellia,with_leaves`
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。
   - `manifest_file` — 您要建立的清單檔案的名稱。
   - (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

from datetime import datetime, timezone
import argparse
import logging
import csv
```

```
import os
import json

"""
Purpose
Amazon Rekognition Custom Labels model example used in the service documentation.
Shows how to create an image-level (classification) manifest file from a CSV file.
You can specify multiple image level labels per image.
CSV file format is
image,label,label,..
If necessary, use the bucket argument to specify the S3 bucket folder for the
 images.
https://docs.aws.amazon.com/rekognition/latest/customlabels-dg/md-gt-cl-
transform.html
"""

logger = logging.getLogger(__name__)


def check_duplicates(csv_file, deduplicated_file, duplicates_file):
    """
    Checks for duplicate images in a CSV file. If duplicate images
    are found, deduplicated_file is the deduplicated CSV file - only the first
    occurence of a duplicate is recorded. Other duplicates are recorded in
 duplicates_file.
    :param csv_file: The source CSV file.
    :param deduplicated_file: The deduplicated CSV file to create. If no duplicates
 are found
    this file is removed.
    :param duplicates_file: The duplicate images CSV file to create. If no
 duplicates are found
    this file is removed.
    :return: True if duplicates are found, otherwise false.
    """

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

    duplicates_found = False

    # Find duplicates.
    with open(csv_file, 'r', newline='', encoding="UTF-8") as f,\
            open(deduplicated_file, 'w', encoding="UTF-8") as dedup,\
            open(duplicates_file, 'w', encoding="UTF-8") as duplicates:
```

```
        reader = csv.reader(f, delimiter=',')
        dedup_writer = csv.writer(dedup)
        duplicates_writer = csv.writer(duplicates)

        entries = set()
        for row in reader:
            # Skip empty lines.
            if not ''.join(row).strip():
                continue

            key = row[0]
            if key not in entries:
                dedup_writer.writerow(row)
                entries.add(key)
            else:
                duplicates_writer.writerow(row)
                duplicates_found = True

    if duplicates_found:
        logger.info("Duplicates found check %s", duplicates_file)

    else:
        os.remove(duplicates_file)
        os.remove(deduplicated_file)

    return duplicates_found


def create_manifest_file(csv_file, manifest_file, s3_path):
    """
    Reads a CSV file and creates a Custom Labels 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 S3 path to the folder that contains the images.
    """
    logger.info("Processing CSV file %s", csv_file)

    image_count = 0
    label_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 CSV file.
        for row in image_classifications:
            source_ref = str(s3_path)+row[0]

            image_count += 1

            # Create JSON for image source ref.
            json_line = {}
            json_line['source-ref'] = source_ref

            # Process each image level label.
            for index in range(1, len(row)):
                image_level_label = row[index]

                # Skip empty columns.
                if image_level_label == '':
                    continue
                label_count += 1

               # Create the JSON line metadata.
                json_line[image_level_label] = 1
                metadata = {}
                metadata['confidence'] = 1
                metadata['job-name'] = 'labeling-job/' + image_level_label
                metadata['class-name'] = image_level_label
                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[f'{image_level_label}-metadata'] = metadata

                # Write the image JSON Line.
            output_file.write(json.dumps(json_line))
            output_file.write('\n')

    output_file.close()
    logger.info("Finished creating manifest file %s\nImages: %s\nLabels: %s",
                manifest_file, image_count, label_count)

    return image_count, label_count
```

```
def add_arguments(parser):
    """
    Adds 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 S3 bucket and folder path for the images."
        " If not supplied, column 1 is assumed to include the 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 = ''

        # Create file names.
        csv_file = args.csv_file
        file_name = os.path.splitext(csv_file)[0]
        manifest_file = f'{file_name}.manifest'
        duplicates_file = f'{file_name}-duplicates.csv'
        deduplicated_file = f'{file_name}-deduplicated.csv'

        # Create manifest file, if there are no duplicate images.
        if check_duplicates(csv_file, deduplicated_file, duplicates_file):
            print(f"Duplicates found. Use {duplicates_file} to view duplicates "
```

```
                  f"and then update {deduplicated_file}. ")
            print(f"{deduplicated_file} contains the first occurence of a
 duplicate. "
                  "Update as necessary with the correct label information.")
            print(f"Re-run the script with {deduplicated_file}")
        else:
            print("No duplicates found. Creating manifest file.")

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

            print(f"Finished creating manifest file: {manifest_file} \n"
                  f"Images: {image_count}\nLabels: {label_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. 如果您打算使用測試資料集，請重複步驟 1 至 3，為測試資料集建立清單檔案。
5. 如有必要，請將影像複製到您在 CSV 檔案的資料欄 1 中指定的 Amazon S3 儲存貯體路徑 (或在 `--s3_path` 命令列中指定)。您可以使用下列 AWS S3 命令。

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

6. 將清單檔案上傳到您要用於存放清單檔案的 Amazon S3 儲存貯體。

> **Note**
>
> 請確保 Amazon Rekognition 自訂標籤可存取清單檔案 JSON Lines 的 `source-ref` 欄位中參考的 Amazon S3 儲存貯體。如需詳細資訊，請參閱 存取外部 Amazon S3 儲存貯體。如果您的 Ground Truth 任務將影像存放在 Amazon Rekognition 自訂標籤主控台儲存貯體中，則不需要新增權限。

7. 請遵循 使用 SageMaker Ground Truth 清單文件創建數據集（控制台） 中的指示，使用上傳的資訊清單檔案建立資料集。對於步驟 8，請在 .manifest 檔案位置，輸入清單檔案位置的 Amazon S3

URL。如果您使用 AWS SDK，請執行 使用 SageMaker Ground Truth 資訊清單檔案 (SDK) 建立資料集。

現有的資料集

如果您先前已建立資料集，則可以將其內容複製到新的資料集。若要使用 AWS SDK 從現有資料集建立資料集，請參閱 使用現有資料集 (SDK) 建立資料集。

使用現有 Amazon Rekognition 自訂標籤資料集 (主控台) 建立資料集

1. 在 https://console.aws.amazon.com/rekognition/ 開啟 Amazon Rekognition 主控台。
2. 選擇使用自訂標籤。
3. 選擇 啟動。
4. 在左側導覽窗格中，選擇專案。
5. 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。
6. 選擇建立資料集。建立資料集頁面即會顯示。
7. 在開始設定中，選擇從單一資料集開始或從訓練資料集開始。若要建立更高品質的模型，我們建議您從個別的訓練和測試資料集開始。

   Single dataset

   a. 在訓練資料集詳細資訊區段中，選擇複製現有的 Amazon Rekognition 自訂標籤資料集。
   b. 在訓練資料集詳細資訊區段的資料集編輯方塊中，輸入或選取您要複製的資料集名稱。
   c. 選擇建立資料集。專案的資料集頁面隨即開啟。

   Separate training and test datasets

   a. 在訓練資料集詳細資訊區段中，選擇複製現有的 Amazon Rekognition 自訂標籤資料集。
   b. 在訓練資料集詳細資訊區段的資料集編輯方塊中，輸入或選取您要複製的資料集名稱。
   c. 在測試資料集詳細資訊區段中，選擇複製現有的 Amazon Rekognition 自訂標籤資料集。
   d. 在測試資料集詳細資訊區段的資料集編輯方塊中，輸入或選取您要複製的資料集名稱。

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

e. 選擇建立資料集。專案的資料集頁面隨即開啟。

8. 如果您需要新增或變更標籤，請執行 標記檔案。

9. 請遵循 訓練模型（控制台） 中的步驟訓練模型。

標記檔案

標籤可識別影像中物件周圍的物件、場景、概念或週框方塊。例如，如果您的資料集包含狗的影像，您可以新增犬種的標籤。

將影像匯入資料集之後，您可能需要對影像新增標籤，或更正標記錯誤的影像。例如，從本機電腦匯入的影像即沒有標記。您可以使用資料集圖庫新增標籤至資料集，並將標籤和週框方塊指派給資料集中的影像。

您在資料集中標記影像的方式會決定 Amazon Rekognition 自訂標籤訓練的模型類型。如需詳細資訊，請參閱 規劃資料集。

主題

- 管理標籤
- 將影像層級標籤指派給影像
- 使用週框方塊標記物件

管理標籤

您可使用 Amazon Rekognition 自訂標籤主控台管理影像。沒有用於管理標籤的特定 API — 當您使用 `CreateDataset` 建立資料集或使用 `UpdateDatasetEntries` 新增更多影像至資料集時，標籤會新增至資料集。

主題

- 管理標籤 (主控台)
- 管理標籤 (SDK)

管理標籤 (主控台)

您可以使用 Amazon Rekognition 自訂標籤主控台來新增、變更標籤或從資料集移除標籤。若要將標籤新增至資料集，您可以新增您建立的新標籤或從 Rekognition 中的現有資料集匯入標籤。

主題

新增標籤 (主控台)

您可以指定要新增至資料集的新標籤。

使用編輯視窗新增標籤

新增標籤 (主控台)

1. 在 https://console.aws.amazon.com/rekognition/ 開啟 Amazon Rekognition 主控台。
2. 選擇使用自訂標籤。
3. 選擇 啟動。
4. 在左側導覽窗格中，選擇專案。
5. 在所有專案頁面上，選擇您要使用的專案。專案的詳細資訊頁面隨即顯示。
6. 如果您要為訓練資料集新增標籤，請選擇訓練索引標籤。否則，請選擇測試索引標籤，將標籤新增至測試資料集。
7. 選擇開始標記以進入標記模式。
8. 在資料集圖庫的標籤區段中，選擇管理標籤，以開啟管理標籤對話方塊。
9. 在編輯方塊中，輸入新標籤名稱。
10. 選擇新增標籤。
11. 重複步驟 9 和 10，直到您建立完所需的標籤為止。
12. 選擇儲存以儲存您新增的標籤。

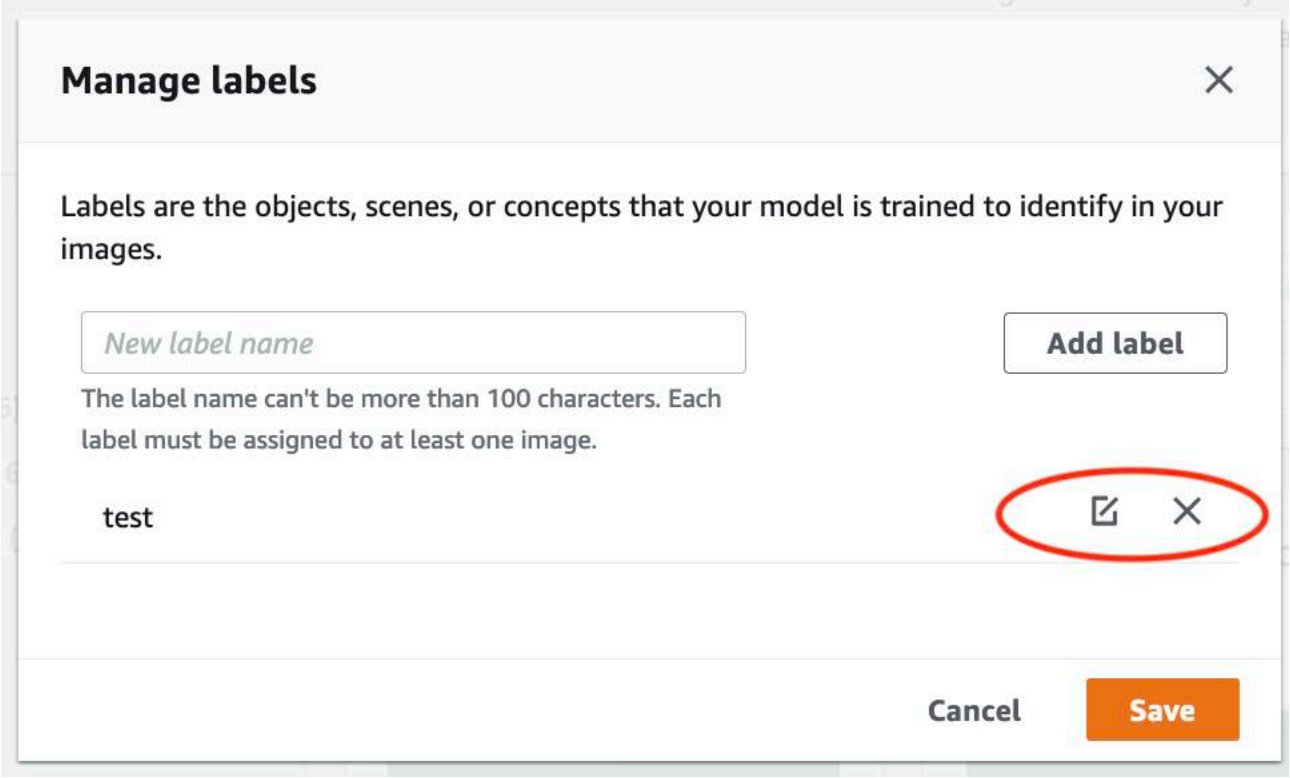
變更和移除標籤 (主控台)

您可以在將標籤新增至資料集後重新命名或移除標籤。您只能移除未指派給任何影像的標籤。

重新命名或移除現有標籤 (主控台)

1. 在 https://console.aws.amazon.com/rekognition/ 開啟 Amazon Rekognition 主控台。
2. 選擇使用自訂標籤。

3. 選擇 啟動。
4. 在左側導覽窗格中，選擇專案。
5. 在所有專案頁面上，選擇您要使用的專案。專案的詳細資訊頁面隨即顯示。
6. 如果您想要變更或刪除訓練資料集中的標籤，請選擇訓練索引標籤。否則，請選擇測試索引標籤，以變更或刪除測試資料集的標籤。
7. 選擇開始標記以進入標記模式。
8. 在資料集圖庫的標籤區段中，選擇管理標籤，以開啟管理標籤對話方塊。
9. 選擇您要編輯或刪除的標籤。



   a. 如果您選擇刪除圖示 (X)，則會從清單中移除標籤。
   b. 如果要變更標籤，請選擇編輯圖示 (鉛筆和紙墊)，然後在編輯方塊中輸入新的標籤名稱。
10. 選擇儲存，以儲存變更。

管理標籤 (SDK)

沒有管理資料集標籤的唯一 API。如果您使用 `CreateDataset`、在清單檔案或複製的資料集中找到的標籤建立資料集，請建立初始的標籤集。如果您使用 `UpdateDatasetEntries` API 新增更多影像，則在項目中找到的新標籤會新增至資料集。如需詳細資訊，請參閱 新增更多影像 (SDK)。若要刪除資料集中的標籤，您必須移除資料集中的所有標籤註釋。

從資料集中刪除標籤

1. 呼叫 `ListDatasetEntries` 以取得資料集項目。如需範例程式碼，請參閱 列出資料集項目 (SDK)。
2. 在檔案中，移除所有標籤註釋。如需詳細資訊，請參閱 清單檔案中的影像層級標籤 及 the section called “清單檔案中的物件本地化”。
3. 使用檔案以透過 `UpdateDatasetEntries` API 更新資料集。如需詳細資訊，請參閱 新增更多影像 (SDK)。

將影像層級標籤指派給影像

您可以使用影像層級標籤來訓練將影像分類為不同類別的模型。影像層級標籤表示影像包含物件、場景或概念。例如，下列影像即顯示河流。如果您的模型將影像分類為包含河流，則需要新增 river 影像層級標籤。如需詳細資訊，請參閱 規劃資料集。

包含影像層級標籤的資料集至少需要定義兩個標籤。每個影像都需要至少一個指派的標籤，以識別影像中的物件、場景或概念。

將影像層級標籤指派給影像 (主控台)

1. 在 https://console.aws.amazon.com/rekognition/ 開啟 Amazon Rekognition 主控台。
2. 選擇使用自訂標籤。
3. 選擇 啟動。
4. 在左側導覽窗格中，選擇專案。
5. 在所有專案頁面上，選擇您要使用的專案。專案的詳細資訊頁面隨即顯示。
6. 在左側導覽窗格中，選擇 [資料集]。
7. 如果您要為訓練資料集新增標籤，請選擇訓練索引標籤。否則，請選擇測試索引標籤，將標籤新增至測試資料集。
8. 選擇開始標記以進入標記模式。
9. 在影像圖庫中，選取您要新增標籤的一或多個影像。您一次只能選取單一頁面上的影像。若要在頁面上選取連續範圍的影像：
    a. 選取範圍中的第一個影像。
    b. 按住 Shift 鍵。
    c. 選取最後一個影像範圍。也會選取第一和第二個影像之間的影像。
    d. 放開 Shift 鍵。
10. 選擇 指派圖像層級標籤。
11. 在「指定影像層級標籤給選取的影像」對話方塊中，選取要指定給一或多個影像的標籤。
12. 選擇 指派，為圖像指派標籤。
13. 重複標記，直到每個影像都以所需的標籤進行註釋。
14. 請選擇儲存變更，以儲存您所做的變更。

指派影像層級標籤 (SDK)

您可以使用 `UpdateDatasetEntries` API 新增或更新指派給影像的影像層級標籤。`UpdateDatasetEntries` 需要一個或多個 JSON Lines。每個 JSON Line 代表一個影像。對於具有影像層級標籤的影像，JSON Line 看起來類似以下內容。

```
{"source-ref":"s3://custom-labels-console-us-east-1-nnnnnnnnnn/gt-job/
manifest/IMG_1133.png","TestCLConsoleBucket":0,"TestCLConsoleBucket-metadata":
```

```
{"confidence":0.95,"job-name":"labeling-job/testclconsolebucket","class-name":"Echo
 Dot","human-annotated":"yes","creation-
date":"2020-04-15T20:17:23.433061","type":"groundtruth/image-classification"}}
```

source-ref 欄位表示影像的位置。JSON Line 也包含指派給影像的影像層級標籤。如需詳細資訊，請參閱 the section called “清單檔案中的影像層級標籤”。

將影像層級標籤指派給影像

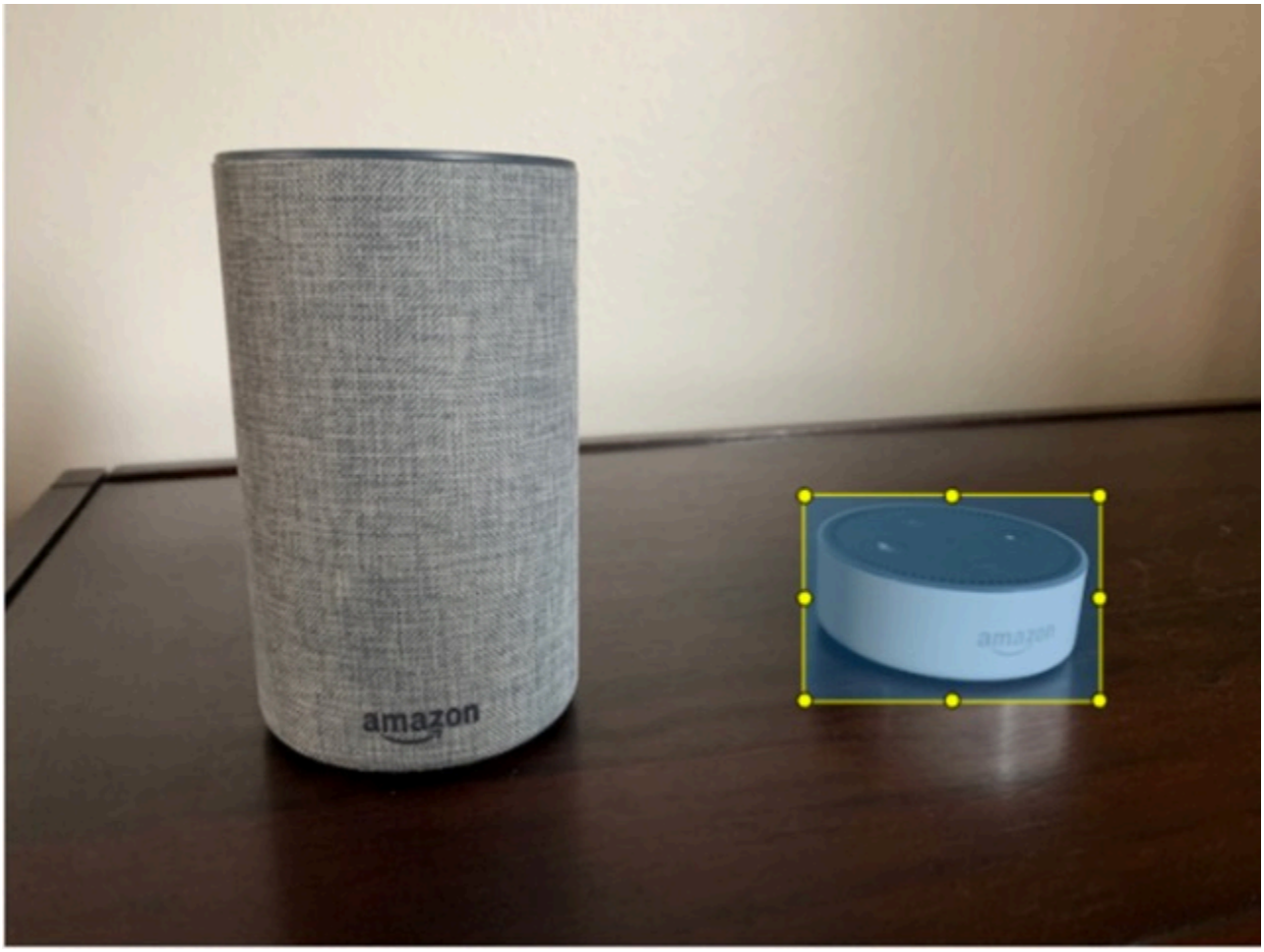
1. 使用 ListDatasetEntries 取得現有影像的 JSON Line。對於 source-ref 欄位，指定要為其指派標籤的影像的位置。如需詳細資訊，請參閱 列出資料集項目 (SDK)。
2. 使用 清單檔案中的影像層級標籤 中的資訊更新上一個步驟中傳回的 JSON Line。
3. 呼叫 UpdateDatasetEntries 以更新影像。如需詳細資訊，請參閱 將更多影像新增至資料集。

使用週框方塊標記物件

如果您希望模型偵測影像中物件的位置，您必須識別物件是什麼，以及物件在影像中的位置。週框方塊是隔離影像中物件的方塊。您可以使用週框方塊來訓練模型，以偵測相同影像中的不同物件。您可以透過將標籤指派給週框方塊來識別物件。

> **Note**
>
> 如果您正在訓練模型以尋找具有影像層級標籤的物件、場景和概念，則不需要執行此步驟。

例如，如果您想要訓練偵測 Amazon Echo Dot 裝置的模型，您可以在影像中的每個 Echo Dot 周圍繪製一個週框方塊，並為週框方塊指派一個名為 Echo Dot 的標籤。下列影像即顯示 Echo Dot 裝置周圍的週框方塊。影像還包含一個沒有週框方塊的 Amazon Echo。



使用週框方塊尋找物件 (主控台)

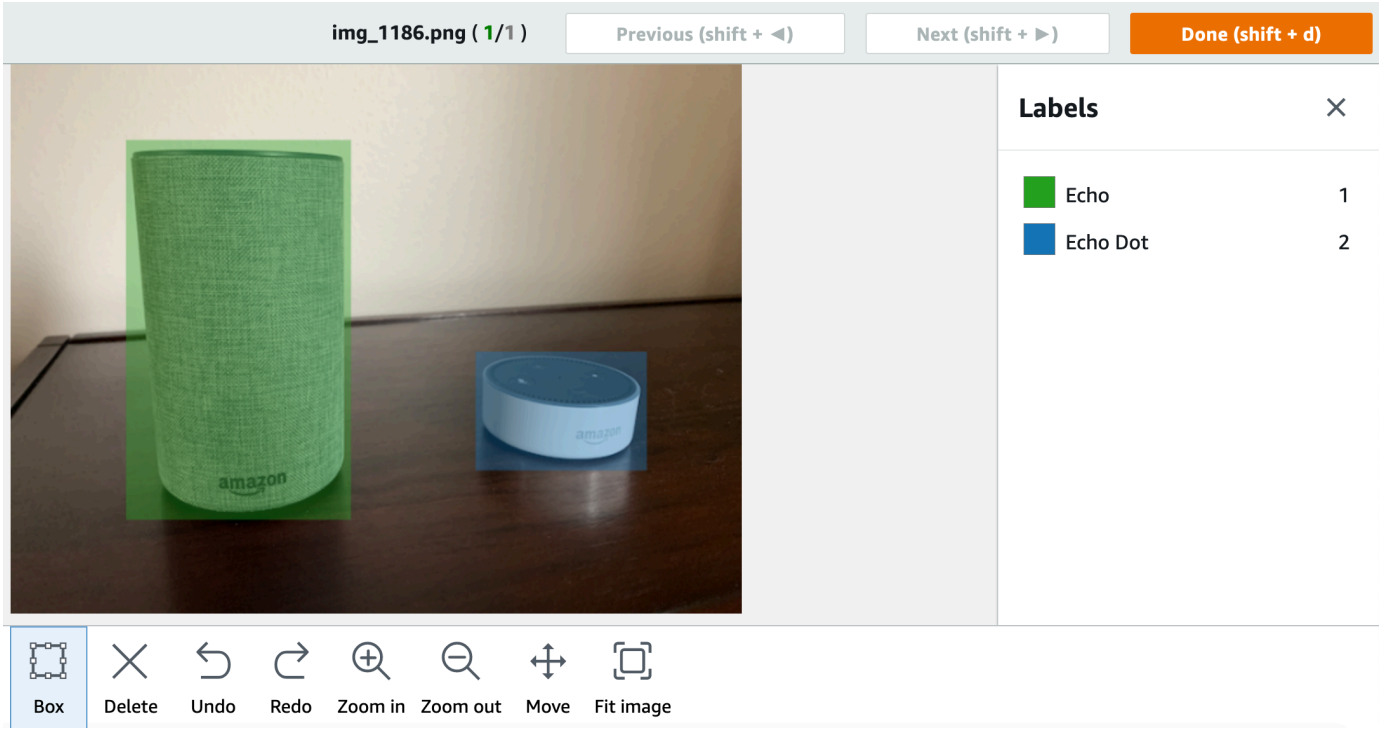
在此程序中，您可以使用主控台繪製影像中物件周圍的週框方塊。您也可以透過將標籤指派給週框方塊來識別物件。

> **Note**
>
> 您無法使用 Safari 瀏覽器對影像新增週框方塊。如需支援的瀏覽器，請參閱 設定 Amazon Rekognition 自訂標籤。

您必須先新增至少一個標籤至資料集，才能新增週框方塊。如需詳細資訊，請參閱 新增標籤 (主控台)。

1. 在 https://console.aws.amazon.com/rekognition/ 開啟 Amazon Rekognition 主控台。
2. 選擇使用自訂標籤。

3. 選擇 啟動。
4. 在左側導覽窗格中，選擇專案。
5. 在所有專案頁面上，選擇您要使用的專案。專案的詳細資訊頁面隨即顯示。
6. 在專案詳細資訊頁面上，選擇標籤影像
7. 如果您想要新增週框方塊到訓練資料集影像，請選擇訓練索引標籤。否則，請選擇測試索引標籤，將週框方塊新增到測試資料集影像。
8. 選擇開始標記以進入標記模式。
9. 在影像圖庫中，選擇您要新增週框方塊的影像。
10. 選擇繪製週框方塊。在顯示週框方塊編輯器之前，會先顯示一系列提示。
11. 在右側的標籤窗格中，選取要指派給週框方塊的標籤。
12. 在繪圖工具中，將指標置於所需物件的左上角區域。
13. 按下滑鼠左鍵並在物件周圍繪製一個方塊。嘗試繪製盡可能靠近物件的週框方塊。
14. 放開滑鼠按鈕。週框方塊會反白顯示。
15. 如果要標記更多影像，請選擇下一步。否則，請選擇完成，以完成標記。



16. 重複步驟 1 至 7，直到您在每個包含物件的影像中建立週框方塊為止。
17. 選擇儲存變更，以儲存您所做的變更。
18. 選擇退出，可退出標記模式。

使用週框方塊尋找物件 (SDK)

您可以使用 UpdateDatasetEntries API 來新增或更新影像的物件位置資訊。UpdateDatasetEntries 需要一或多個 JSON Lines。每個 JSON Line 代表一個影像。對於物件本地化，JSON Line 看起來類似如下內容。

```
{"source-ref": "s3://bucket/images/IMG_1186.png", "bounding-box": {"image_size":
 [{"width": 640, "height": 480, "depth": 3}], "annotations": [{ "class_id": 1,
 "top": 251, "left": 399, "width": 155, "height": 101}, {"class_id": 0, "top": 65,
 "left": 86, "width": 220, "height": 334}]}, "bounding-box-metadata": {"objects":
 [{ "confidence": 1}, {"confidence": 1}], "class-map": {"0": "Echo", "1": "Echo Dot"},
 "type": "groundtruth/object-detection", "human-annotated": "yes", "creation-date":
 "2013-11-18T02:53:27", "job-name": "my job"}}
```

source-ref 欄位表示影像的位置。JSON Line 也包含影像上每個物件標記的週框方塊。如需詳細資訊，請參閱 the section called “清單檔案中的物件本地化”。

將週框方塊指派給影像

1. 使用 ListDatasetEntries 取得現有影像的 JSON Line。對於 source-ref 欄位，指定要為其指派影像層級標籤的影像的位置。如需詳細資訊，請參閱 列出資料集項目 (SDK)。
2. 使用 清單檔案中的物件本地化 中的資訊更新上一個步驟中傳回的 JSON Line。
3. 呼叫 UpdateDatasetEntries 以更新影像。如需詳細資訊，請參閱 將更多影像新增至資料集。

偵錯資料集

在資料集建立期間，可能會發生兩種類型的錯誤 — 終端錯誤和非終端錯誤。終端錯誤可能會停止資料集建立或更新。終端錯誤則不會停止資料集建立或更新。

主題

終端錯誤

終端錯誤有兩種類型：導致資料集建立失敗的檔案錯誤，以及 Amazon Rekognition 自訂標籤從資料集中移除的內容錯誤。如果內容錯誤太多，資料集建立會失敗。

主題

- 終端檔案錯誤
- 終端內容錯誤

終端檔案錯誤

以下是檔案錯誤。呼叫 DescribeDataset 並檢查 Status 和 StatusMessage 欄位，即可取得檔案錯誤的相關資訊。如需範例程式碼，請參閱 描述資料集 (SDK)。

- ERROR_MANIFEST_INACCESSIBLE_OR_UNSUPPORTED_FORMAT
- ERROR_MANIFEST_SIZE_TOO_LARGE.
- ERROR_MANIFEST_ROWS_EXCEEDS_MAXIMUM
- ERROR_INVALID_PERMISSIONS_MANIFEST_S3_BUCKET
- ERROR_TOO_MANY_RECORDS_IN_ERROR
- ERROR_MANIFEST_TOO_MANY_LABELS
- ERROR_INSUFFICIENT_IMAGES_PER_LABEL_FOR_DISTRIBUTE

ERROR_MANIFEST_INACCESSIBLE_OR_UNSUPPORTED_FORMAT

錯誤訊息

清單檔案副檔名或內容無效。

訓練或測試清單檔案沒有副檔名或其內容無效。

修正錯誤 ERROR_MANIFEST_INACCESSIBLE_OR_UNSUPPORTED_FORMAT

- 檢查訓練和測試清單檔案中的下列可能原因。
  - 清單檔案缺少副檔名。按照慣例，檔案副檔名為 .manifest。
  - 找不到清單檔案的 Amazon S3 儲存貯體或金鑰。

ERROR_MANIFEST_SIZE_TOO_LARGE

錯誤訊息

清單檔案大小超過支援的大小上限。

訓練或測試清單檔案大小 (以位元組為單位) 太大。如需詳細資訊，請參閱 Amazon Rekognition 自訂標籤中的準則。清單檔案的 JSON Lines 數目可能少於最大數目，但仍超過檔案大小上限。

您無法使用 Amazon Rekognition 自訂標籤主控台修正錯誤清單檔案大小超過支援的大小上限。

修正錯誤 ERROR_MANIFEST_SIZE_TOO_LARGE

1. 檢查哪些訓練和測試清單檔案超出檔案大小上限。
2. 減少過大的清單檔案中的 JSON Lines 數目。如需詳細資訊，請參閱 建立清單檔案。

ERROR_MANIFEST_ROWS_EXCEEDS_MAXIMUM

錯誤訊息

清單檔案的資料列太多。

其他資訊

清單檔案中的 JSON Lines 數目 (影像數目) 大於允許的限制。影像層級模型和物件位置模型的限制不同。如需詳細資訊，請參閱 Amazon Rekognition 自訂標籤中的準則。

JSON Line 錯誤會被驗證，直到 JSON Lines 數目達到 `ERROR_MANIFEST_ROWS_EXCEEDS_MAXIMUM` 限制為止。

您無法使用 Amazon Rekognition 自訂標籤主控台修正錯誤 `ERROR_MANIFEST_ROWS_EXCEEDS_MAXIMUM`。

修正 **`ERROR_MANIFEST_ROWS_EXCEEDS_MAXIMUM`**

- 減少清單檔案中的 JSON Lines 數目。如需詳細資訊，請參閱 建立清單檔案。

ERROR_INVALID_PERMISSIONS_MANIFEST_S3_BUCKET

錯誤訊息

S3 儲存貯體權限不正確。

Amazon Rekognition 自訂標籤不具有一或多個包含訓練和測試清單檔案的儲存貯體的權限。

您無法使用 Amazon Rekognition 自訂標籤主控台修正此錯誤。

修正錯誤 ERROR_INVALID_PERMISSIONS_MANIFEST_S3_BUCKET

- 檢查包含訓練和測試清單檔案的儲存貯體的權限。如需詳細資訊，請參閱 步驟 2：設定 Amazon Rekognition 自訂標籤主控台權限。

ERROR_TOO_MANY_RECORDS_IN_ERROR

錯誤訊息

清單檔案的終端錯誤太多。

修正 **ERROR_TOO_MANY_RECORDS_IN_ERROR**

- 減少具有終端內容錯誤之 JSON Lines (影像) 的數量。如需詳細資訊，請參閱 終端資訊內容錯誤。

您無法使用 Amazon Rekognition 自訂標籤主控台修正此錯誤。

ERROR_MANIFEST_TOO_MANY_LABELS

錯誤訊息

清單檔案的標籤太多。

其他資訊

清單檔案 (資料集) 中的唯一標籤數目超過允許的限制。如果分割訓練資料集以建立測試資料集，則標籤數量會在分割後確定。

修正 ERROR_MANIFEST_TOO_MANY_LABELS (主控台)

- 從資料集中移除標籤。如需詳細資訊，請參閱 管理標籤。標籤會自動從資料集中的影像和週框方塊中移除。

修正 ERROR_MANIFEST_TOO_MANY_LABELS (JSON Line)

- 具有影像層級 JSON Lines 的清單檔案 — 如果影像具有單一標籤，請移除使用所需標籤的影像的 JSON Lines。如果 JSON Line 包含多個標籤，請僅移除所需標籤的 JSON 物件。如需詳細資訊，請參閱 對影像新增多個影像層級標籤。

具有物件位置 JSON Lines 的清單檔案 — 移除要移除之標籤的週框方塊和相關聯的標籤資訊。針對包含所需標籤的每個 JSON Line 執行此操作。您需要從 `class-map` 陣列和 `objects` 和 `annotations` 陣列中的對應物件移除標籤。如需詳細資訊，請參閱 清單檔案中的物件本地化。

ERROR_INSUFFICIENT_IMAGES_PER_LABEL_FOR_DISTRIBUTE

錯誤訊息

清單檔案沒有足夠的已標記影像來分發資料集。

當 Amazon Rekognition 自訂標籤分割訓練資料集以建立測試資料集時，即會發生資料集分佈。您也可以透過呼叫 `DistributeDatasetEntries` API 來分割資料集。

修正錯誤 ERROR_MANIFEST_TOO_MANY_LABELS

- 將更多已標記影像新增至訓練資料集

終端內容錯誤

以下是終端內容錯誤。在資料集建立期間，會從資料集中移除具有終端內容錯誤的影像。資料集仍可用於訓練。如果內容錯誤太多，資料集/更新會失敗。與資料集作業相關的終端內容錯誤不會顯示在主控台中，也不會從 `DescribeDataset` 或其他 API 傳回。如果您發現資料集缺少影像或註釋，請檢查資料集清單檔案是否有下列問題：

- JSON Line 的長度太長。長度上限為 100,000 個字元。
- JSON Line 中缺少 `source-ref` 值。
- JSON Line 中 `source-ref` 值的格式無效。
- JSON Line 的內容無效。
- `source-ref` 欄位會顯示多次的值。影像在資料集中只能被參考一次。

如需 `source-ref` 欄位的相關資訊，請參閱 建立清單檔案。

非終端錯誤

以下是資料集建立或更新期間可能發生的非終端錯誤。這些錯誤可能會使整個 JSON Line 無效，或使 JSON Line 中的註釋失效。如果 JSON Line 出現錯誤，則不會用於訓練。如果 JSON Line 中的註釋出

現錯誤，JSON Line 仍會用於訓練，但沒有中斷的註釋。如需 JSON Lines 的詳細資訊，請參閱 建立清單檔案。

您可以從主控台並呼叫 `ListDatasetEntries` API 來存取非終端錯誤。如需詳細資訊，請參閱 列出資料集項目 (SDK)。

訓練期間也會傳回下列錯誤。建議您在訓練模型之前先修正這些錯誤。如需更多詳細資訊，請參閱 非終端 JSON 行驗證錯誤。

- 錯誤標籤屬性
- 錯誤 _ 無效 _ 標籤屬性 _ 格式
- 錯誤 _ 無效 _ 標籤屬性 _ 中繼資料格式
- 有效的標籤屬性錯誤
- 錯誤 _ 無效 _ 邊界 _ 方塊
- 錯誤 _ 無效影像尺寸
- 錯誤 _ 邊界 _ 方塊 _ 太小
- 錯誤有效註釋
- 錯誤 _ 遺失 _ 邊界 _ 信心
- 錯誤 _ 缺少類別 _ 地圖識別碼
- 錯誤 _ 多個 _ 彈出 _ 方塊
- 不支援的 _ 使用案例類型
- 錯誤 _ 無效 _ 標籤名稱 _ 長度

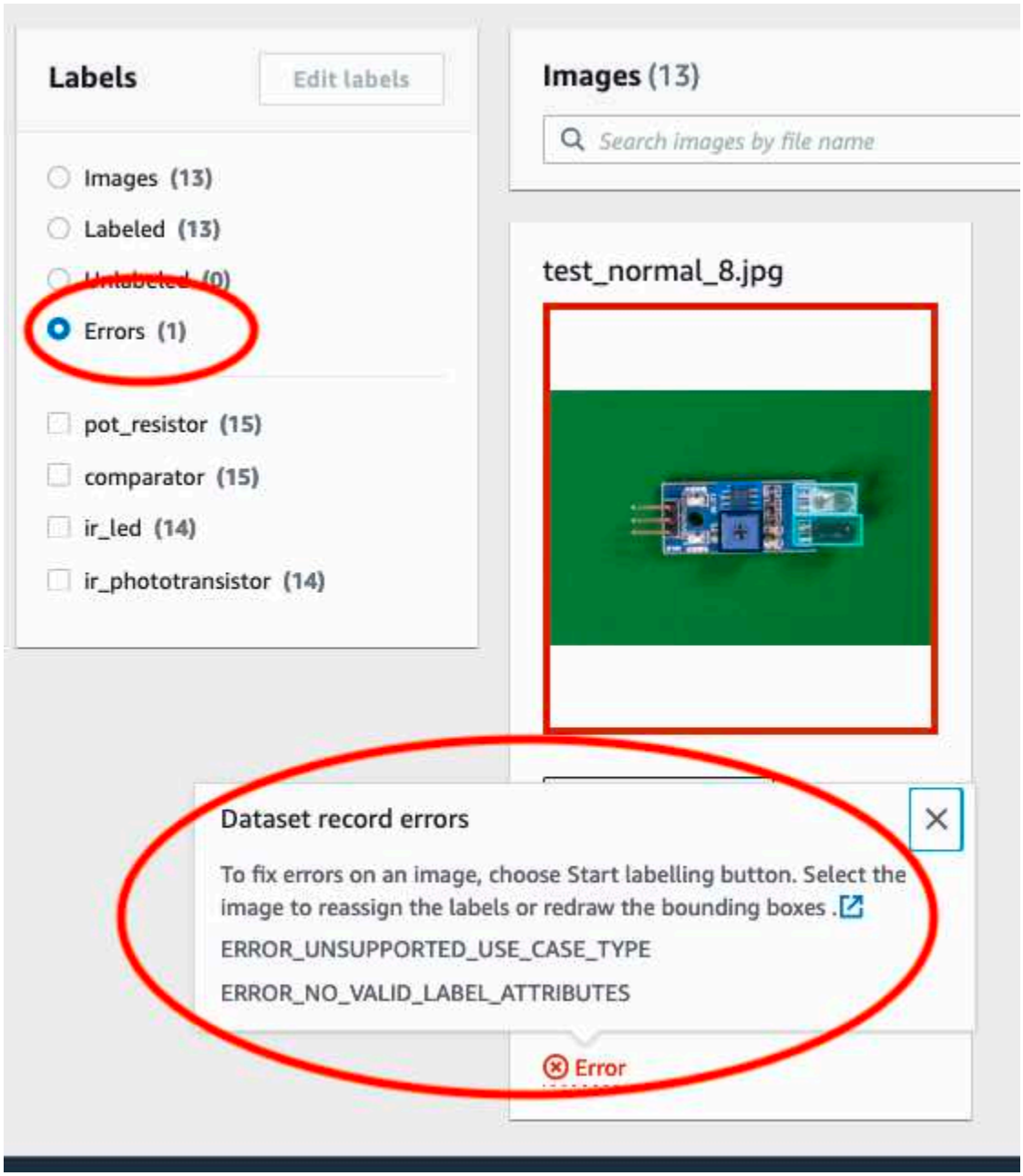
存取非終端錯誤

您可以使用主控台來找出資料集中的哪些影像具有非終端錯誤。您也可以呼叫，呼叫 `ListDatasetEntries` API 以取得錯誤訊息。如需詳細資訊，請參閱 列出資料集項目 (SDK)。

存取非終端錯誤 (主控台)

1. 在 https://console.aws.amazon.com/rekognition/ 開啟 Amazon Rekognition 主控台。
2. 選擇使用自訂標籤。
3. 選擇 啟動。
4. 在左側導覽窗格中，選擇專案。
5. 在所有專案頁面上，選擇您要使用的專案。專案的詳細資訊頁面隨即顯示。

6. 如果您想要檢視訓練資料集中的非終端錯誤，請選擇訓練索引標籤。否則，請選擇測試索引標籤，以檢視測試資料集中的非終端錯誤。
7. 在資料集圖庫的標籤區段中，選擇錯誤。資料集圖庫會經過篩選，只顯示發生錯誤的影像。
8. 選擇影像下方的錯誤以查看錯誤代碼。使用 非終端 JSON 行驗證錯誤 中的資訊來修正錯誤。



訓練 Amazon Rekognition 自訂標籤模型

您可以使用 Amazon Rekognition 自訂標籤主控台或 Amazon Rekognition 自訂標籤 API 來訓練模型。如果模型訓練失敗，請使用中的資訊偵錯失敗的模型訓練來尋找失敗的原因。

> **Note**
>
> 您需支付成功訓練模型所需的時間費用。訓練通常需要 30 分鐘到 24 小時才能完成。如需詳細資訊，請參閱訓練時數。

每次訓練模型時，都會建立模型的新版本。Amazon Rekognition 自訂標籤會為模型建立名稱，該名稱是專案名稱和建立模型時間戳記的組合。

為了訓練您的模型，Amazon Rekognition 自訂標籤會複製您的來源訓練和測試影像。依預設，複製的映像會使用 AWS 擁有和管理的金鑰進行靜態加密。您也可以選擇使用自己的使用自己的使用AWS KMS key。如果您使用自己的 KMS 金鑰，您需要對 KMS 金鑰具備下列權限。

- 公里：CreateGrant
- 公里：DescribeKey

如需詳細資訊，請參閱 AWS Key Management Service 概念。您的來源影像不受影響。

您可以使用 KMS 伺服器端加密 (SSE-KMS) 來加密 Amazon S3 儲存貯體中的訓練和測試映像，然後再由 Amazon Rekognition 自訂標籤複製這些映像。若要允許 Amazon Rekognition 自訂標籤存取您的映像檔，您的AWS帳戶需要下列 KMS 金鑰的許可。

- 公里：GenerateDataKey
- kms:Decrypt

如需詳細資訊，請參閱使用「伺服器端加密搭配存放在 AWS Key Management Service (SSE-KMS) 的 KMS」來保護資料。

訓練模型後，您可以評估其效能並進行改進。如需詳細資訊，請參閱改善訓練有素的 Amazon Rekognition 自訂標籤模型。

如需其他模型工作 (例如為模型貼標籤)，請參閱管理 Amazon Rekognition 自訂標籤模型。

主題

- 訓練模型（控制台）
- 訓練模型 (SDK)

訓練模型（控制台）

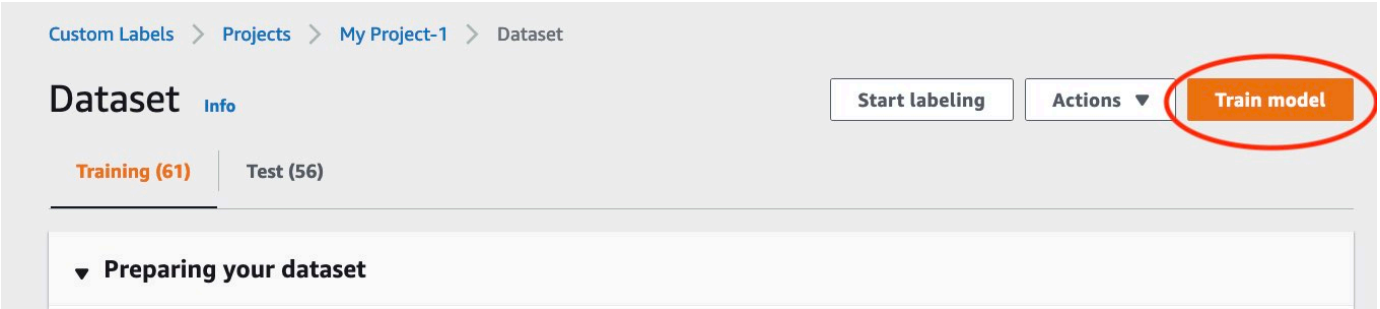
您可以使用 Amazon Rekognition 自訂標籤主控台來訓練模型。

訓練需要具備訓練資料集和測試資料集的專案。如果您的專案沒有測試資料集，Amazon Rekognition 自訂標籤主控台會在訓練期間分割訓練資料集，以便為您的專案建立一個資料集。選擇的影像是具有代表性的取樣，不會用於訓練資料集。我們建議您只在您沒有可使用的替代測試資料集時拆分訓練資料集。分割訓練資料集可減少可用於訓練的影像數量。

> **Note**
>
> 系統會根據訓練模型所需的時間付費。如需詳細資訊，請參閱訓練時數。

訓練您的模型（控制台）

1. 開啟亞馬遜重新認知主控台，網址為 https://console.aws.amazon.com/rekognition/。
2. 選擇「使用自訂標籤」。
3. 在左側導覽窗格中選擇 Project (專案)。
4. 在「專案」頁面中，選擇包含您要訓練之模型的專案。
5. 在 [專案] 頁面上，選擇 [訓練模型]。



6. (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. 在 [影像資料加密] 中選擇 [自訂加密設定 (進階)]。
   b. 在加密 .aws_kms_key 時，請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。
7. (選用) 如果希望新增標籤至模型中新增標籤至模型，請執行以下操作：

a. 在「標籤」區段中，選擇「新增標籤」。

b. 輸入下列：

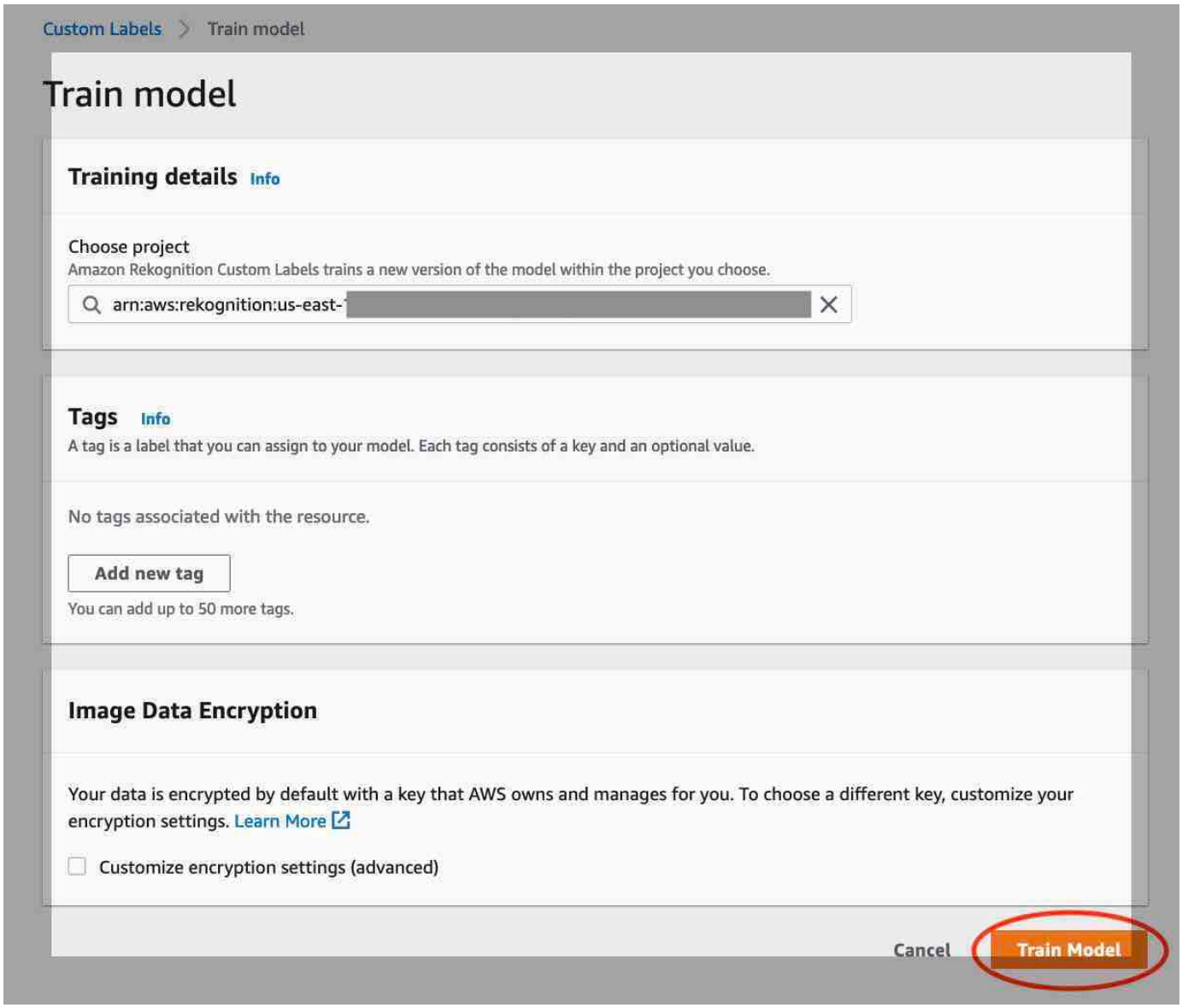
   i. 金鑰中的金鑰名稱。

   ii. 值中鍵的值。

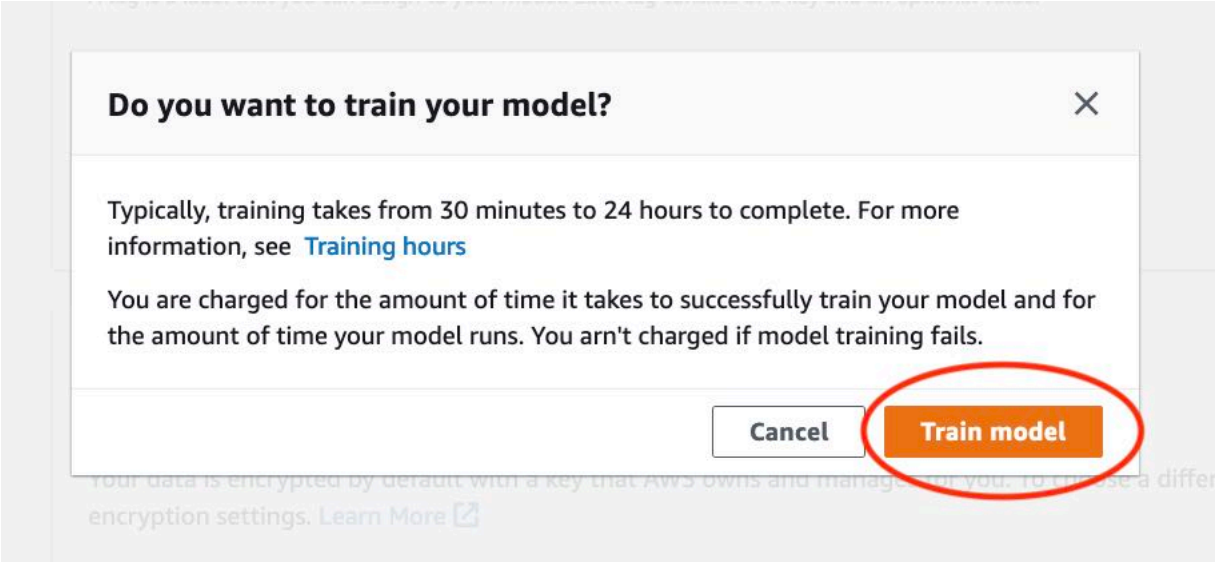
c. 若要新增更多標籤，請重複步驟 6a 和 6b。

d. (選用) 如果希望移除標籤，請在您要移除的標籤旁邊選擇 Remove (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。

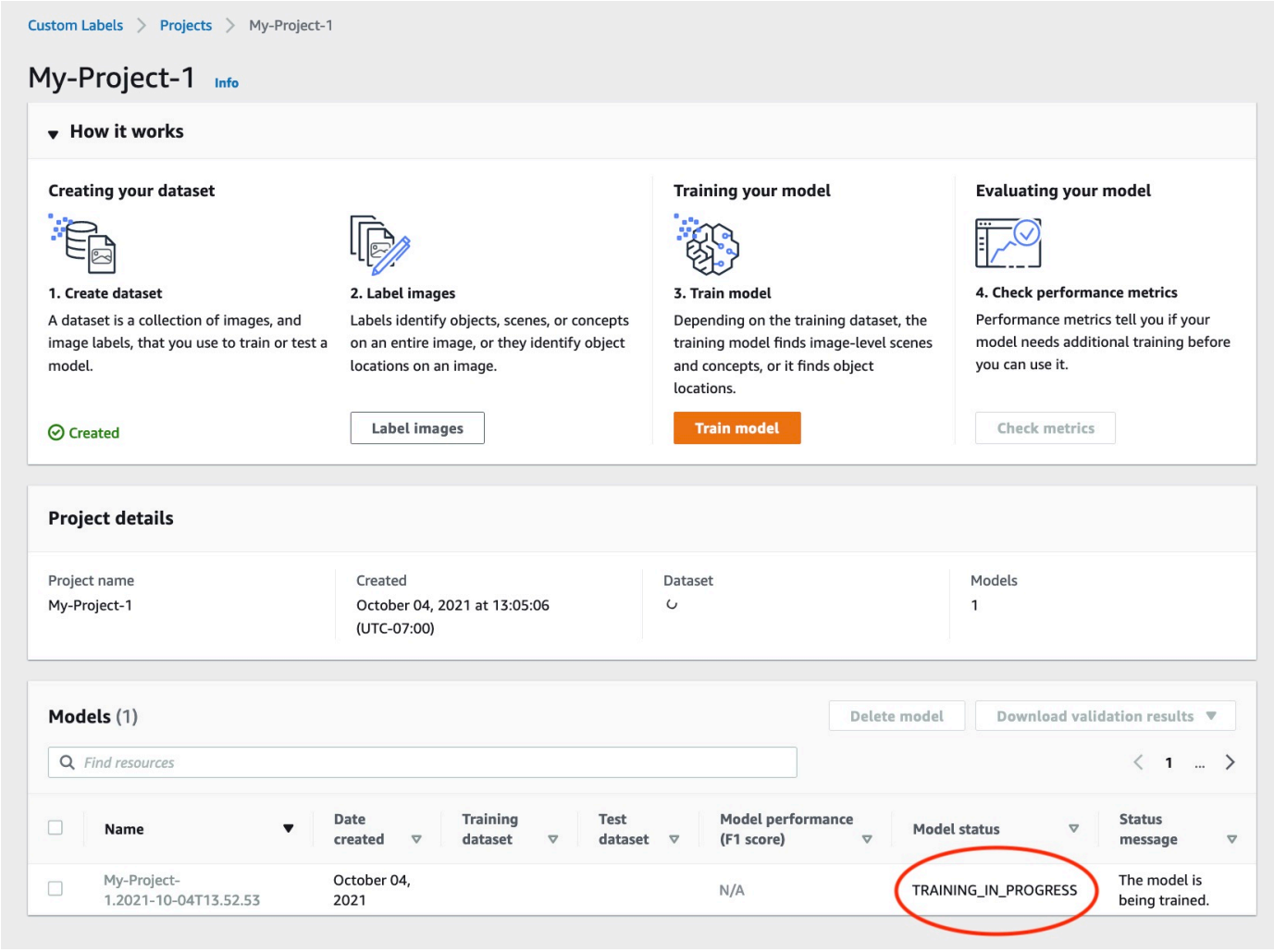
8. 在 [訓練模型] 頁面上，選擇 [訓練模型]。專案的 Amazon Resource Name (ARN) 應在選擇專案編輯方塊中。如果沒有，請輸入專案的 ARN。



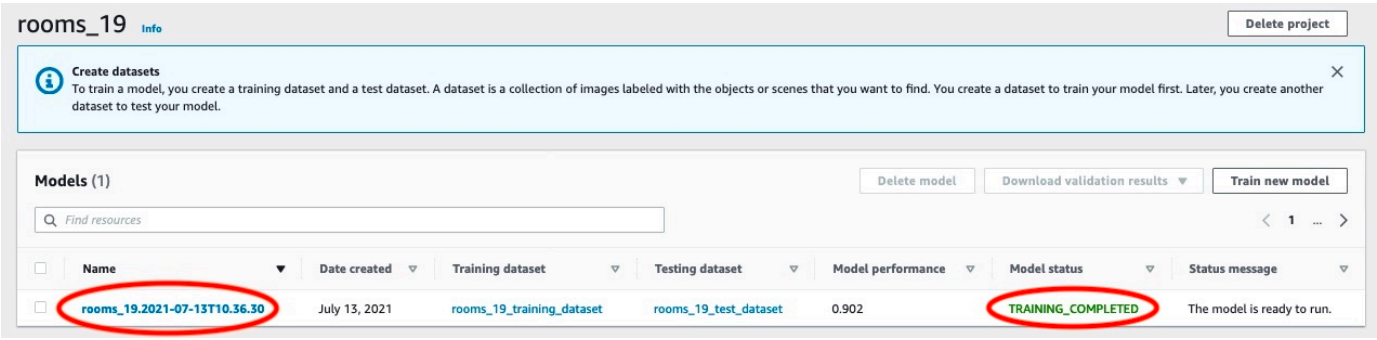
9. 在 [您要訓練您的模型嗎？ 」對話方塊中選擇訓練模型」。



10. 在專案頁面的「模型」區段中，您可以檢查訓練進行中的Model Status欄目前狀態。訓練模型需要一段時間才能完成。



11. 訓練完成後，選擇模型名稱。模型狀態為「訓練 _ 已完成」時，訓練即完成。如果訓練失敗，請閱讀偵錯失敗的模型訓練。



12. 下一步：評估您的模型。如需詳細資訊，請參閱「改善訓練有素的 Amazon Rekognition 自訂標籤模型」。

訓練模型 (SDK)

您可以透過呼叫來訓練模型CreateProjectVersion。若要訓練模型，需要下列資訊：

- Name (名稱)-模型版本的唯一名稱稱稱稱稱稱稱稱
- 專案 ARN — 管理模型的專案的 Amazon Resource Name (ARN)。
- 訓練結果位置 — 放結果的 Amazon S3 位置。您可以使用與主控台 Amazon S3 儲存貯體相同的位置，也可以選擇不同的位置。我們建議您選擇不同的位置，因為這可讓您設定許可，並避免使用 Amazon Rekognition 自訂標籤主控台導致訓練輸出的可能命名衝突。

訓練使用與專案相關聯的訓練和測試資料集。如需詳細資訊，請參閱管理資料集。

> **Note**
>
> 或者，您可以指定專案外部的訓練和測試資料集資訊清單檔案。如果您在使用外部資訊清單檔案訓練模型後開啟主控台，Amazon Rekognition 自訂標籤會使用用於訓練的最後一組資訊清單檔案來為您建立資料集。您無法再透過指定外部資訊清單檔案來訓練專案的模型版本。如需詳細資訊，請參閱CreatePrjectVersion。

來自的回應`CreateProjectVersion`是 ARN，您可以在後續要求中用來識別模型版本。您也可以使用 ARN 保護模型版本。如需詳細資訊，請參閱保護亞馬遜重新認知自訂標籤專案。

訓練模型版本需要一段時間才能完成。本主題中的 Python 和 Java 範例使用服務員來等待訓練完成。服務員是一種實用程序方法，用於輪詢特定狀態發生。或者，您可以通過調用來獲取培訓的當前狀態`DescribeProjectVersions`。當`Status`欄位值為時，就會完成訓練`TRAINING_COMPLETED`。訓練完成後，您可以檢閱評估結果來評估模型的品質。

訓練模型 (SDK)

下列範例顯示如何使用與專案相關聯的訓練和測試資料集來訓練模型。

若要訓練模型 (SDK)

1. 若您尚未這樣做，請安裝AWS CLI並設定和AWS SDK。如需詳細資訊，請參閱步驟 4：設定 AWS CLI 和 AWS SDKs。
2. 使用下列範例程式碼來訓練專案。

AWS CLI

下列範例會建立模型。訓練資料集會分割以建立測試資料集。取代下列項目：

- my_project_arn使用專案的 Amazon Resource Name (ARN)。
- version_name使用您選擇的唯一版本名稱。
- output_bucket使用「Amazon S3 儲體的名稱，會在 Amazon S3 儲體中保護 Amazon Rekognition S3 儲體的名稱，會在該儲體中保護訓練結果。
- output_folder以儲存訓練結果的資料夾名稱。
- (選用參數)--kms-key-id 搭配存放 AWS Key Management Service 客戶主金鑰的 ID。

```
aws rekognition create-project-version \
  --project-arn project_arn \
  --version-name version_name \
  --output-config '{"S3Bucket":"output_bucket", "S3KeyPrefix":"output_folder"}'
 \
  --profile custom-labels-access
```

Python

下列範例會建立模型。提供下列命令列引數：

- project_arn— 專案的 Amazon Resource Name (ARN)。
- version_name— 您選擇的模型的唯一版本名稱。
- output_bucket— Amazon S3 儲體的名稱，Amazon S3 儲體的名稱，會在該儲體中保護訓練結果的 Amazon S3 儲體名稱，會在該儲體的
- output_folder— 儲存訓練結果的資料夾名稱。

(可選) 提供以下指令行參數，以將標籤貼附至模型：

- tag— 您希望連接到模型的您選擇的標籤名稱，您希望連接到模型上。
- tag_value標籤值。

```
#Copyright 2023 Amazon.com, Inc. or its affiliates. All Rights Reserved.
```

```
#PDX-License-Identifier: MIT-0 (For details, see https://github.com/awsdocs/
amazon-rekognition-custom-labels-developer-guide/blob/master/LICENSE-
SAMPLECODE.)


import argparse
import logging
import json
import boto3

from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)

def train_model(rek_client, project_arn, version_name, output_bucket,
 output_folder, tag_key, tag_key_value):
    """
    Trains an Amazon Rekognition Custom Labels model.
    :param rek_client: The Amazon Rekognition Custom Labels Boto3 client.
    :param project_arn: The ARN of the project in which you want to train a
 model.
    :param version_name: A version for the model.
    :param output_bucket: The S3 bucket that hosts training output.
    :param output_folder: The path for the training output within output_bucket
    :param tag_key: The name of a tag to attach to the model. Pass None to
 exclude
    :param tag_key_value: The value of the tag. Pass None to exclude

    """

    try:
        #Train the model

        status=""
        logger.info("training model version %s for project %s",
            version_name, project_arn)


        output_config = json.loads(
            '{"S3Bucket": "'
            + output_bucket
            + '", "S3KeyPrefix": "'
            + output_folder
            + '" }  '
```

```
        )

        tags={}

        if tag_key is not None and tag_key_value is not None:
            tags = json.loads(
                '{"' + tag_key + '":"' + tag_key_value + '"}'
            )


        response=rek_client.create_project_version(
            ProjectArn=project_arn,
            VersionName=version_name,
            OutputConfig=output_config,
            Tags=tags
        )

        logger.info("Started training: %s", response['ProjectVersionArn'])

        # Wait for the project version training to complete.

        project_version_training_completed_waiter =
 rek_client.get_waiter('project_version_training_completed')
        project_version_training_completed_waiter.wait(ProjectArn=project_arn,
        VersionNames=[version_name])


        # Get the completion status.

 describe_response=rek_client.describe_project_versions(ProjectArn=project_arn,
            VersionNames=[version_name])
        for model in describe_response['ProjectVersionDescriptions']:
            logger.info("Status: %s", model['Status'])
            logger.info("Message: %s", model['StatusMessage'])
            status=model['Status']


        logger.info("finished training")

        return response['ProjectVersionArn'], status

    except ClientError as err:
        logger.exception("Couldn't create model: %s", err.response['Error']
['Message'] )
```

```
        raise

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

    parser.add_argument(
        "project_arn", help="The ARN of the project in which you want to train a
 model"
    )

    parser.add_argument(
        "version_name", help="A version name of your choosing."
    )

    parser.add_argument(
        "output_bucket", help="The S3 bucket that receives the training
 results."
    )

    parser.add_argument(
        "output_folder", help="The folder in the S3 bucket where training
 results are stored."
    )

    parser.add_argument(
        "--tag_name",  help="The name of a tag to attach to the model",
 required=False
    )

    parser.add_argument(
        "--tag_value",  help="The value for the tag.", 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()

        print(f"Training model version {args.version_name} for project
 {args.project_arn}")

        # Train the model.
        session = boto3.Session(profile_name='custom-labels-access')
        rekognition_client = session.client("rekognition")

        model_arn, status=train_model(rekognition_client,
            args.project_arn,
            args.version_name,
            args.output_bucket,
            args.output_folder,
            args.tag_name,
            args.tag_value)


        print(f"Finished training model: {model_arn}")
        print(f"Status: {status}")


    except ClientError as err:
        logger.exception("Problem training model: %s", err)
        print(f"Problem training model: {err}")
    except Exception as err:
        logger.exception("Problem training model: %s", err)
        print(f"Problem training model: {err}")


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

Java V2

下列範例會訓練模型。提供下列命令列引數：

- `project_arn`— 專案的 Amazon Resource Name (ARN)。

- version_name— 您選擇的模型的唯一版本名稱。
- output_bucket— Amazon S3 儲體的名稱，Amazon S3 儲體的名稱，會在該儲體中保護訓練結果的 Amazon S3 儲體名稱，會在該儲體的
- output_folder— 儲存訓練結果的資料夾名稱。

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

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import
 software.amazon.awssdk.services.rekognition.model.CreateProjectVersionRequest;
import
 software.amazon.awssdk.services.rekognition.model.CreateProjectVersionResponse;
import
 software.amazon.awssdk.services.rekognition.model.DescribeProjectVersionsRequest;
import
 software.amazon.awssdk.services.rekognition.model.DescribeProjectVersionsResponse;
import software.amazon.awssdk.services.rekognition.model.OutputConfig;
import
 software.amazon.awssdk.services.rekognition.model.ProjectVersionDescription;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import software.amazon.awssdk.services.rekognition.waiters.RekognitionWaiter;

import java.util.Optional;
import java.util.logging.Level;
import java.util.logging.Logger;

public class TrainModel {

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

    public static String trainMyModel(RekognitionClient rekClient, String
 projectArn, String versionName,
            String outputBucket, String outputFolder) {
```

```
       try {

           OutputConfig outputConfig =
OutputConfig.builder().s3Bucket(outputBucket).s3KeyPrefix(outputFolder).build();

           logger.log(Level.INFO, "Training Model for project {0}",
projectArn);
           CreateProjectVersionRequest createProjectVersionRequest =
CreateProjectVersionRequest.builder()

.projectArn(projectArn).versionName(versionName).outputConfig(outputConfig).build();

           CreateProjectVersionResponse response =
rekClient.createProjectVersion(createProjectVersionRequest);

           logger.log(Level.INFO, "Model ARN: {0}",
response.projectVersionArn());
           logger.log(Level.INFO, "Training model...");

           // wait until training completes

           DescribeProjectVersionsRequest describeProjectVersionsRequest =
DescribeProjectVersionsRequest.builder()
                   .versionNames(versionName)
                   .projectArn(projectArn)
                   .build();

           RekognitionWaiter waiter = rekClient.waiter();

           WaiterResponse<DescribeProjectVersionsResponse> waiterResponse =
waiter

.waitUntilProjectVersionTrainingCompleted(describeProjectVersionsRequest);

           Optional<DescribeProjectVersionsResponse> optionalResponse =
waiterResponse.matched().response();

           DescribeProjectVersionsResponse describeProjectVersionsResponse =
optionalResponse.get();

           for (ProjectVersionDescription projectVersionDescription :
describeProjectVersionsResponse
                   .projectVersionDescriptions()) {
```

```
                System.out.println("ARN: " +
projectVersionDescription.projectVersionArn());
                System.out.println("Status: " +
projectVersionDescription.statusAsString());
                System.out.println("Message: " +
projectVersionDescription.statusMessage());
            }

            return response.projectVersionArn();

        } catch (RekognitionException e) {
            logger.log(Level.SEVERE, "Could not train model: {0}",
e.getMessage());
            throw e;
        }

    }

    public static void main(String args[]) {

        String versionName = null;
        String projectArn = null;
        String projectVersionArn = null;
        String bucket = null;
        String location = null;

        final String USAGE = "\n" + "Usage: " + "<project_name> <version_name>
<output_bucket> <output_folder>\n\n" + "Where:\n"
                + "   project_arn - The ARN of the project that you want to use.
\n\n"
                + "   version_name - A version name for the model.\n\n"
                + "   output_bucket - The S3 bucket in which to place the
training output. \n\n"
                + "   output_folder - The folder within the bucket that the
training output is stored in. \n\n";

        if (args.length != 4) {
            System.out.println(USAGE);
            System.exit(1);
        }

        projectArn = args[0];
        versionName = args[1];
        bucket = args[2];
```

```
        location = args[3];

        try {

            // Get the Rekognition client.
            RekognitionClient rekClient = RekognitionClient.builder()
            .credentialsProvider(ProfileCredentialsProvider.create("custom-
labels-access"))
            .region(Region.US_WEST_2)
            .build();


            // Train model
            projectVersionArn = trainMyModel(rekClient, projectArn, versionName,
 bucket, location);

            System.out.println(String.format("Created model: %s for Project ARN:
 %s", projectVersionArn, projectArn));

            rekClient.close();

        } catch (RekognitionException rekError) {
            logger.log(Level.SEVERE, "Rekognition client error: {0}",
 rekError.getMessage());
            System.exit(1);
        }

    }

}
```

3. 如果訓練失敗，請閱讀偵錯失敗的模型訓練。

偵錯失敗的模型訓練

您可能會在模型訓練期間遇到錯誤。Amazon Rekognition 自訂標籤會在主控台和回應中報告訓練錯誤。DescribeProjectVersions

錯誤可能是終端（培訓無法繼續），或者是非終端（培訓可以繼續）。對於與訓練和測試資料集內容相關的錯誤，您可以下載驗證結果 (資訊清單摘要，以及訓練與測試驗證資訊清單)。使用驗證結果中的錯誤代碼來尋找本節中的進一步資訊。本節還提供資訊清單檔案錯誤 (在驗證資訊清單檔案內容之前發生的終端機錯誤) 的資訊。

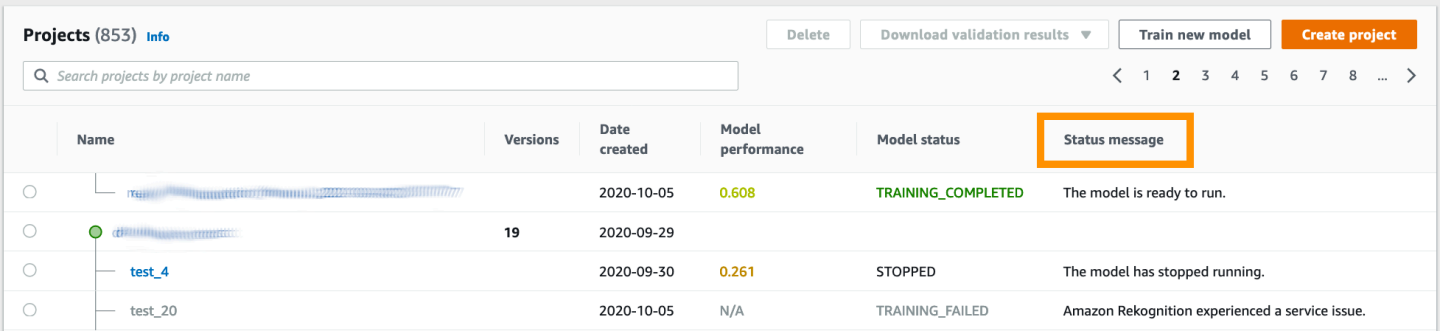
> **Note**
>
> 清單是用於存儲數據集的內容的文件。

您可以使用 Amazon Rekognition 自訂標籤主控台修正一些錯誤。其他錯誤可能會要求您更新訓練或測試資訊清單檔案。您可能需要進行其他變更，例如 IAM 許可。如需詳細資訊，請參閱個別錯誤的文件。

終端機錯誤

終端機錯誤會停止模型的訓練。終端訓練錯誤有 3 種類別 — 服務錯誤、資訊清單檔案錯誤和資訊清單內容錯誤。

在主控台中，Amazon Rekognition 自訂標籤會在專案頁面的「狀態訊息」欄中顯示模型的終端機錯誤。



如果您使用 AWS SDK，則可以通過檢查響應來DescribeProjectVersions了解終端機清單文件錯誤或終端機清單內容錯誤是否發生了。在此情況下，`Status`值為`TRAINING_FAILED`且`StatusMessage`欄位包含錯誤。

服務錯誤

當 Amazon Rekognition 遇到服務問題且無法繼續訓練時，就會發生終端機服務錯誤。例如，Amazon Rekognition 自訂標籤所依賴的另一項服務失敗。Amazon Rekognition 自訂標籤會在主控台中報告服務錯誤，因為 Amazon Rekognition 遇到服務問題。如果您使用 AWS SDK，則訓練期間發生的服務錯誤會由 CreateProjectVersionand 引發為`InternalServerError`例外狀況DescribeProjectVersions。

如果發生服務錯誤，請重試模型的訓練。如果培訓持續失敗，請聯絡 AWS Support，並包含任何回報服務錯誤的錯誤資訊。

終端機清單檔錯誤

資訊清單檔案錯誤是指在訓練和測試資料集中發生在檔案層級或跨多個檔案的終端機錯誤。在驗證訓練和測試資料集的內容之前，會偵測到資訊清單檔案錯誤。資訊清單檔案錯誤會防止報告非終端驗證錯誤。例如，一個空的訓練清單文件生成一個清單文件是空的錯誤。由於檔案為空，因此無法報告非終端 JSON Line 驗證錯誤。資訊清單摘要也不會建立。

您必須先修正資訊清單檔案錯誤，才能訓練模型。

以下列出資訊清單檔案錯誤。

- 資訊清單副檔名或內容無效。
- 清單文件是空的。
- 資訊清單檔案大小超過支援的大小上限。
- 無法寫入至輸出 S3 儲存貯體。
- S3 儲存貯體許可不正確。

終端資訊內容錯誤

資訊清單內容錯誤是與資訊清單中的內容相關的終端機錯誤。例如，如果您收到錯誤資訊清單檔案包含每個標籤的標籤影像不足以執行自動分割，訓練無法完成，因為訓練資料集中沒有足夠的標籤影像無法建立測試資料集。

除了在控制台和來自的響應中報告錯誤之外`DescribeProjectVersions`，還會在清單摘要中報告錯誤以及任何其他終端機清單內容錯誤。如需詳細資訊，請參閱了解資訊清單摘要。

非終端 JSON 行錯誤也會在單獨的訓練和測試驗證結果清單中報告。Amazon Rekognition 自訂標籤找到的非終端 JSON 行錯誤不一定與停止訓練的資訊清單內容錯誤相關。如需詳細資訊，請參閱了解培訓和測試驗證結果清單。

您必須先修正資訊清單內容錯誤，才能訓練模型。

以下是資訊清單內容錯誤的錯誤訊息。

- 資訊清單檔案包含太多無效資料列。
- 資訊清單檔案包含來自多個 S3 儲存貯體的映像。
- 映像 S3 儲存貯體的擁有者識別碼無效。
- 資訊清單檔案每個標籤包含不足的標籤影像，無法執行自動分割。

- 資訊清單檔案的標籤太少。
- 資訊清單檔案有太多的標籤。
- 訓練和測試資訊清單檔案之間的標籤重疊小於 {}%。
- 資訊清單檔案的可用標籤太少。
- 訓練和測試資訊清單檔案之間的可用標籤重疊小於 {}%。
- 無法從 S3 儲存貯體複製映像。

非終端 JSON 行驗證錯誤

JSON 線路驗證錯誤是非終端錯誤，不需要 Amazon Rekognition 自訂標籤即可停止訓練模型。

JSON 行驗證錯誤不會顯示在控制台中。

在訓練和測試資料集中，JSON Line 代表單一影像的訓練或測試資訊。JSON Line 中的驗證錯誤 (例如無效的影像) 會在訓練和測試驗證資訊清單中報告。Amazon Rekognition 自訂標籤會使用資訊清單中的其他有效 JSON 行完成訓練。如需詳細資訊，請參閱了解培訓和測試驗證結果清單。如需驗證規則的詳細資訊，請參閱清單檔案的驗證規則。

> **Note**
>
> 如果 JSON 行錯誤太多，則訓練會失敗。

我們建議您也修正非終端 JSON Line 錯誤錯誤，因為這些錯誤可能會造成 future 錯誤或影響您的模型訓練。

Amazon Rekognition 自訂標籤可以產生下列非終端 JSON 行驗證錯誤。

- 來源參考索引鍵遺失。
- 來源參考值的格式無效。
- 找不到標籤屬性。
- 標籤屬性 {} 的格式無效。
- 標籤屬性中繼資料的格式無效。
- 找不到有效的標籤屬性。
- 一或多個邊界方框缺少置信度值。

- 類別對應中缺少其中一個類別 ID。
- JSON 行的格式無效。
- 影像無效。檢查 S3 路徑和/或映像屬性。
- 邊界方框具有關閉框架值。
- 邊界方框的高度和寬度太小。
- 邊界方框超過允許的最大值。
- 找不到有效的註釋。

了解資訊清單摘要

資訊清單摘要包含以下資訊。

- 驗證期間終端資訊內容錯誤遇到的錯誤資訊。
- 訓練和測試資料集非終端 JSON 行驗證錯誤中的錯誤位置資訊。
- 錯誤統計資料，例如在訓練和測試資料集中找到的無效 JSON 行總數。

如果沒有，則會在訓練期間建立資訊清單摘要終端機清單檔錯誤。若要取得資訊清單摘要檔案的位置，請參閱。 取得驗證結果

> **Note**
>
> 資訊清單摘要中不會報告服務錯誤和資訊清單檔案錯誤。如需詳細資訊，請參閱終端機錯誤。

如需特定資訊清單內容錯誤的資訊，請參閱終端資訊內容錯誤。

清單摘要檔案格式

清單文件有 2 個部分，`statistics`和`errors`.

統計資訊

`statistics`包含訓練和測試資料集中錯誤的相關資訊。

- `training`— 在訓練資料集中找到的統計資料和錯誤。
- `testing`— 在測試數據集中發現的統計信息和錯誤。

errors陣列中的物件包含資訊清單內容錯誤的錯誤碼和訊息。

error_line_indices陣列包含有錯誤的訓練或測試資訊清單中每個 JSON 行的行號。如需詳細資訊，請參閱修正訓練錯誤。

錯誤

跨越訓練和測試資料集的錯誤。例如，錯誤_不足_可用_標籤_重疊當沒有足夠的可用標籤與訓練和測試資料集重疊時，就會發生這種情況。

```
{
    "statistics": {
        "training":
            {
                "use_case": String, # Possible values are IMAGE_LEVEL_LABELS,
 OBJECT_LOCALIZATION and NOT_DETERMINED
                "total_json_lines": Number,   # Total number json lines (images) in the
  training manifest.
                "valid_json_lines": Number,   # Total number of JSON Lines (images)
 that can be used for training.
                "invalid_json_lines": Number, # Total number of invalid JSON Lines.
 They are not used for training.
                "ignored_json_lines": Number, # JSON Lines that have a valid schema but
 have no annotations. The aren't used for training and aren't counted as invalid.
                "error_json_line_indices": List[int], # Contains a list of line numbers
 for JSON line errors in the training dataset.
                "errors": [
                    {
                        "code": String, # Error code for a training manifest content
 error.
                        "message": String # Description for a training manifest content
 error.
                    }
                ]
            },
        "testing":
            {
                "use_case": String, # Possible values are IMAGE_LEVEL_LABELS,
 OBJECT_LOCALIZATION and NOT_DETERMINED
                "total_json_lines": Number, # Total number json lines (images) in the
 manifest.
                "valid_json_lines": Number,  # Total number of JSON Lines (images) that
 can be used for testing.
```

```
                "invalid_json_lines": Number, # Total number of invalid JSON Lines.
 They are not used for testing.
                "ignored_json_lines": Number, # JSON Lines that have a valid schema but
 have no annotations. They aren't used for testing and aren't counted as invalid.
                "error_json_line_indices": List[int], # contains a list of error record
 line numbers in testing dataset.
                "errors": [
                    {
                        "code": String,   # # Error code for a testing manifest content
 error.
                        "message": String # Description for a testing manifest content
 error.
                    }
                ]
            }
    },
    "errors": [
        {
            "code": String, # # Error code for errors that span the training and
 testing datasets.
            "message": String # Description of the error.
        }
    ]
}
```

資訊清單摘要

下列範例是部分資訊清單摘要，顯示終端機資訊清單內容錯誤 (錯誤 _ 無效 _ 列資訊清單)。error_json_line_indices陣列包含相應訓練或測試驗證資訊清單中非終端 JSON Line 錯誤的行號。

```
{
    "errors": [],
    "statistics": {
        "training": {
            "use_case": "NOT_DETERMINED",
            "total_json_lines": 301,
            "valid_json_lines": 146,
            "invalid_json_lines": 155,
            "ignored_json_lines": 0,
            "errors": [
                {
                    "code": "ERROR_TOO_MANY_INVALID_ROWS_IN_MANIFEST",
```

```
                    "message": "The manifest file contains too many invalid rows."
                }
            ],
            "error_json_line_indices": [
                15,
                16,
                17,
                22,
                23,
                24,
                 .
                 .
                 .
                 .
                300
            ]
        },
        "testing": {
            "use_case": "NOT_DETERMINED",
            "total_json_lines": 15,
            "valid_json_lines": 13,
            "invalid_json_lines": 2,
            "ignored_json_lines": 0,
            "errors": [],
            "error_json_line_indices": [
                13,
                15
            ]
        }
    }
}
```

了解培訓和測試驗證結果清單

在訓練期間，Amazon Rekognition 自訂標籤會建立驗證結果資訊清單，以保留非終端 JSON 行錯誤。驗證結果資訊清單是訓練和測試資料集的複本，並新增了錯誤資訊。訓練完成後，您可以存取驗證資訊清單。如需詳細資訊，請參閱取得驗證結果。Amazon Rekognition 自訂標籤也會建立資訊清單摘要，其中包含 JSON 行錯誤的概觀資訊，例如錯誤位置和 JSON 行錯誤計數。如需詳細資訊，請參閱了解資訊清單摘要。

> **Note**
>
> 只有在沒終端機清單檔錯誤有的情況下，才會建立驗證結果 (訓練與測試驗證結果資訊清單和資訊清單摘要)。

清單包含 JSON 行數據集中的每個圖像。在驗證結果清單中，JSON 行錯誤信息添加到發生錯誤的 JSON 行。

JSON 行錯誤是與單個圖像相關的非終端錯誤。非終端驗證錯誤可能會使整個 JSON 行或只有一部分無效。例如，如果 JSON 行中參照的影像不是 PNG 或 JPG 格式，就會發生錯錯誤 _ 無效影像誤，而且整個 JSON 行會從訓練中排除。使用其他有效的 JSON 行繼續訓練。

在 JSON 行中，錯誤可能意味著 JSON 行可以用於培訓。例如，如果與標籤相關聯的四個邊界框之一的左邊值為負值，則仍會使用其他有效的邊界方塊來訓練模型。針對無效的邊界方塊 (錯誤 _ 無效 _ 邊界 _ 方塊) 傳回 JSON 行錯誤資訊。在此範例中，會將錯誤資訊新增至發生錯誤的`annotation`物件。

警告錯誤，例如警告無註解，不會用於訓練，並在資訊清單摘要中計為忽略的 JSON 行 (`ignored_json_lines`)。如需詳細資訊，請參閱了解資訊清單摘要。此外，忽略的 JSON 行不會計入訓練和測試的 20% 錯誤閾值。

如需有關特定非終端資料驗證錯誤的資訊，請參閱非終端 JSON 行驗證錯誤。

> **Note**
>
> 如果資料驗證錯誤太多，則會停止訓練，並在資訊清單摘要中報告錯誤 _ 無效 _ 列資訊清單終端機錯誤。

如需更正 JSON 行錯誤的詳細資訊，請參閱修正訓練錯誤。

JSON 行錯誤格式

Amazon Rekognition 自訂標籤會將非終端機驗證錯誤資訊新增至映像層級和物件本地化格式 JSON 行。如需詳細資訊，請參閱the section called “建立清單檔案”。

影像層級錯誤

下面的例子顯示了在圖像級 JSON 線的Error數組。有兩組錯誤。與標籤屬性中繼資料 (在此範例中為運動中繼資料) 相關的錯誤，以及與影像相關的錯誤。錯誤包括錯誤代碼（代碼），錯誤消息（消息）。如需詳細資訊，請參閱清單檔案中的影像層級標籤。

```
{
    "source-ref": String,
    "sport": Number,
    "sport-metadata": {
        "class-name": String,
        "confidence": Float,
        "type": String,
        "job-name": String,
        "human-annotated": String,
        "creation-date": String,
        "errors": [
            {
                "code": String, # error codes for label
                "message": String # Description and additional contextual details of
 the error
            }
        ]
    },
    "errors": [
        {
            "code": String, # error codes for image
            "message": String # Description and additional contextual details of the
 error
        }
    ]
}
```

物件本地化錯誤

下面的例子顯示了對象本地化 JSON 行中的錯誤數組。JSON 行包含下列 JSON 行區段中欄位的Errors陣列資訊。每個Error物件都包含錯誤碼和錯誤訊息。

- 標籤屬性 — 標籤屬性欄位的錯誤。請參閱範例bounding-box中的。
- 註釋 — 註釋錯誤 (邊界框) 儲存在 label 屬性內的annotations陣列中。
- 標籤屬性中繼資料 — 標籤屬性中繼資料的錯誤。請參閱範例bounding-box-metadata中的。
- image — 與標籤屬性、註釋和標籤屬性中繼資料欄位無關的錯誤。

如需詳細資訊，請參閱清單檔案中的物件本地化。

```
{
    "source-ref": String,
    "bounding-box": {
        "image_size": [
            {
                "width": Int,
                "height": Int,
                "depth":Int,
            }
        ],
        "annotations": [
            {
                "class_id": Int,
                "left": Int,
                "top": Int,
                "width": Int,
                "height": Int,
                "errors": [   # annotation field errors
                    {
                        "code": String, # annotation field error code
                        "message": String # Description and additional contextual
 details of the error
                    }
                ]
            }
        ],
        "errors": [ #label attribute field errors
            {
                "code": String, # error code
                "message": String # Description and additional contextual details of
 the error
            }
        ]
    },
    "bounding-box-metadata": {
        "objects": [
            {
                "confidence": Float
            }
        ],
        "class-map": {
            String: String
```

```
        },
        "type": String,
        "human-annotated": String,
        "creation-date": String,
        "job-name": String,
        "errors": [  #metadata field errors
            {
                "code": String, # error code
                "message": String # Description and additional contextual details of
the error
            }
        ]
    },
   "errors": [  # image errors
        {
            "code": String, # error code
            "message": String # Description and additional contextual details of the
error
        }
    ]
}
```

JSON 行錯誤示例

下列物件當地語系化 JSON 行 (格式化以供讀取) 顯示錯錯誤 _ 邊界 _ 方塊 _ 太小誤。在此範例中，邊界方框尺寸 (高度和寬度) 不大於 1 x 1。

```
{
    "source-ref": "s3://bucket/Manifests/images/199940-1791.jpg",
    "bounding-box": {
        "image_size": [
            {
                "width": 3000,
                "height": 3000,
                "depth": 3
            }
        ],
        "annotations": [
            {
                "class_id": 1,
                "top": 0,
                "left": 0,
                "width": 1,
```

```
                "height": 1,
                "errors": [
                    {
                        "code": "ERROR_BOUNDING_BOX_TOO_SMALL",
                        "message": "The height and width of the bounding box is too
 small."
                    }
                ]
            },
            {
                "class_id": 0,
                "top": 65,
                "left": 86,
                "width": 220,
                "height": 334
            }
        ]
    },
    "bounding-box-metadata": {
        "objects": [
            {
                "confidence": 1
            },
            {
                "confidence": 1
            }
        ],
        "class-map": {
            "0": "Echo",
            "1": "Echo Dot"
        },
        "type": "groundtruth/object-detection",
        "human-annotated": "yes",
        "creation-date": "2019-11-20T02:57:28.288286",
        "job-name": "my job"
    }
}
```

取得驗證結果

驗證結果包含和的錯誤終端資訊內容錯誤資訊非終端 JSON 行驗證錯誤。有三個驗證結果檔案。

- 訓練資料集資訊清單檔案的副本，其中新增了 JSON 行錯誤資訊。

- 測試數據集清單文件的副本-添加了 JSON 行錯誤錯誤信息的測試數據集清單文件的副本。
- 清單摘要 .json — 在訓練和測試資料集中找到的資訊清單內容錯誤和 JSON 行錯誤的摘要。如需詳細資訊，請參閱了解資訊清單摘要。

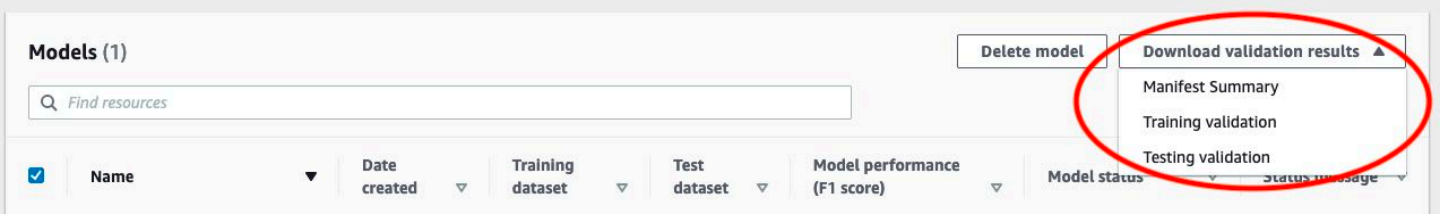
如需有關訓練和測試驗證資訊清單內容的資訊，請參閱偵錯失敗的模型訓練。

> **Note**
>
> - 只有在訓練期間未產生驗證結果時，才終端機清單檔錯誤會建立驗證結果。
> - 如果在驗證訓練和測試資訊清單之後發生服務錯誤，則會建立驗證結果，但來自的回應DescribeProjectVersions不包括驗證結果檔案位置。

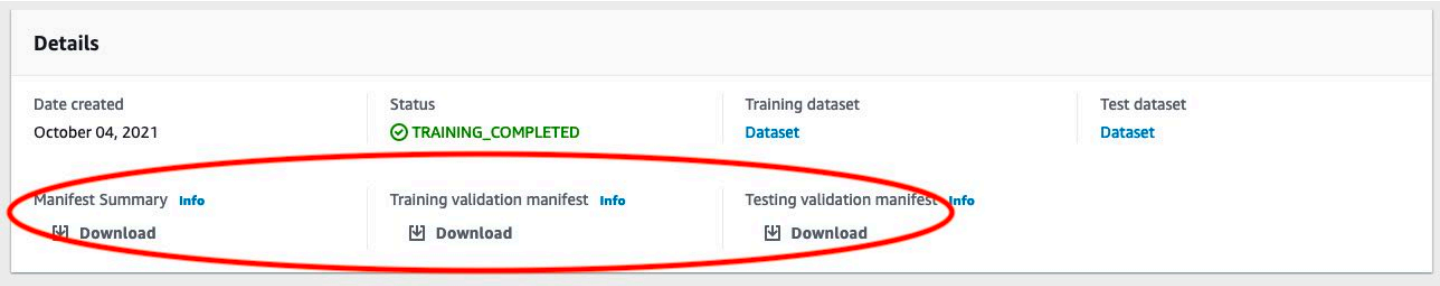
訓練完成或失敗後，您可以使用 Amazon Rekognition 自訂標籤主控台下載驗證結果，或透過呼叫 API 取得 Amazon S3 儲存貯體位置。DescribeProjectVersions

取得驗證結果 (主控台)

如果您使用主控台訓練模型，您可以從專案的模型清單中下載驗證結果，如下圖所示。



您也可以從模型的詳細資訊頁面存取下載驗證結果。



如需詳細資訊，請參閱訓練模型（控制台）。

取得驗證結果 (SDK)

模型訓練完成後，Amazon Rekognition 自訂標籤會將驗證結果存放在訓練期間指定的 Amazon S3 儲存貯體中。訓練完成後，您可以呼叫 DescribeProjectVersionsAPI 來取得 S3 儲存貯體位置。若要訓練模型，請參閱訓練模型 (SDK)。

會針ValidationData對訓練資料集 (TrainingDataResult) 和測試資料集 (TestingDataResult) 傳回物件。資訊清單摘要會在中傳回`ManifestSummary`。

取得 Amazon S3 儲存貯體位置後，您可以下載驗證結果。如需詳細資訊，請參閱如何從 S3 儲存貯體下載物件？ 。您也可以使用此GetObject操作。

若要取得驗證資料 (SDK)

1. 如果您尚未這樣做，請先完成安裝AWS CLI和設定 AWS SDK。如需詳細資訊，請參閱步驟 4：設定 AWS CLI 和 AWS SDKs。
2. 使用以下範例來取得驗證結果的位置。

   Python

   `project_arn`使用包含模型的專案的 Amazon Resource Name (ARN) 取代。如需詳細資訊，請參閱管理 Amazon Rekognition 自訂標籤專案。`version_name`使用模型版本的名稱取代。如需詳細資訊，請參閱訓練模型 (SDK)。

```
import boto3
import io
from io import BytesIO
import sys
import json


def describe_model(project_arn, version_name):

    client=boto3.client('rekognition')

    response=client.describe_project_versions(ProjectArn=project_arn,
        VersionNames=[version_name])

    for model in response['ProjectVersionDescriptions']:
        print(json.dumps(model,indent=4,default=str))

def main():
```

```
    project_arn='project_arn'
    version_name='version_name'

    describe_model(project_arn, version_name)

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

3. 在程式輸出中，請注意TestingDataResult和TrainingDataResult物件中的Validation欄位。資訊清單摘要位於中ManifestSummary。

修正訓練錯誤

您可以使用資訊清單摘要來識別終端資訊內容錯誤並在訓練期間非終端 JSON 行驗證錯誤遇到。您必須修正資訊清單內容錯誤。建議您也修正非終端 JSON 行錯誤。如需特定錯誤的詳細資訊，請參閱非終端 JSON 行驗證錯誤和終端資訊內容錯誤。

您可以修正用於訓練的訓練或測試資料集。或者，您可以在訓練和測試驗證資訊清單檔案中進行修正，並使用它們來訓練模型。

完成修正後，您需要匯入更新的資訊清單並重新訓練模型。如需詳細資訊，請參閱建立清單檔案。

下列程序說明如何使用資訊清單摘要來修正終端機資訊清單內容錯誤。此程序也會示範如何在訓練和測試驗證資訊清單中尋找及修正 JSON Line 錯誤。

修正 Amazon Rekognition 自訂標籤訓練錯誤

1. 下載驗證結果檔案。文件名是訓練清單與驗證 .json，測試清單與驗證. JSON 和清單匯總的 .json。如需詳細資訊，請參閱取得驗證結果。
2. 打開清單摘要文件（清單摘要 .json）。
3. 修正資訊清單摘要中的任何錯誤。如需詳細資訊，請參閱了解資訊清單摘要。
4. 在資訊清單摘要中，遍歷error_line_indices陣列training並修正相應 JSON 行號中training_manifest_with_validation.json的錯誤。如需詳細資訊，請參閱the section called “了解培訓和測試驗證結果清單”。
5. 遍歷數error_line_indices組testing並在相應的 JSON 行號中testing_manifest_with_validation.json修復錯誤。
6. 使用驗證資訊清單檔案做為訓練和測試資料集來重新訓練模型。如需詳細資訊，請參閱the section called “訓練模型”。

如果您使用 AWS SDK 並選擇修正訓練或測試驗證資料資訊清單檔案中的錯誤，請使用驗證資料清單檔案在TrainingData和TestingData輸入參數中的位置CreateProjectVersion。如需詳細資訊，請參閱訓練模型 (SDK)。

JSON 行錯誤優先順序

首先檢測到以下 JSON 行錯誤。如果發生任何這些錯誤，JSON 行錯誤的驗證將停止。您必須先修正這些錯誤，才能修正任何其他 JSON 行錯誤

- 缺少 _ 來源 _ 參考
- 錯誤 _ 無效 _ 來源 _ 參考格式
- 錯誤標籤屬性
- 錯誤 _ 無效 _ 標籤屬性 _ 格式
- 錯誤 _ 無效 _ 標籤屬性 _ 中繼資料格式
- 錯誤 _ 遺失 _ 邊界 _ 信心
- 錯誤 _ 缺少類別 _ 地圖識別碼
- 錯誤 _ 無效 _ J SON_ 行

終端機清單檔錯誤

本主題說明終端機清單檔錯誤。資訊清單檔錯誤沒有關聯的錯誤代碼。當發生終端清單文件錯誤時，不會創建驗證結果清單。如需詳細資訊，請參閱了解資訊清單摘要。終端機資訊清單錯誤會阻止報告非終端 JSON 行驗證錯誤.

資訊清單副檔名或內容無效。

訓練或測試資訊清單檔案沒有副檔名或其內容無效。

修復錯誤清單文件擴展名或內容無效。

- 檢查訓練和測試資訊清單檔案中的下列可能原因。
  - 資訊清單檔案缺少副檔名。按照慣例，文件擴展名是.manifest.
  - 找不到資訊清單檔案的 Amazon S3 儲存貯體或金鑰。

清單文件是空的。

用於訓練的訓練或測試資訊清單檔案存在，但它是空的。資訊清單檔案需要用於訓練和測試的每個影像的 JSON 行。

要修復錯誤清單文件是空的。

1. 檢查哪些培訓或測試清單是空的。
2. 將 JSON 行添加到空清單文件中。如需詳細資訊，請參閱建立清單檔案。或者，使用主控台來建立新的資料集。如需詳細資訊，請參閱the section called “建立包含影像的資料集”。

資訊清單檔案大小超過支援的大小上限。

訓練或測試資訊清單檔案大小 (以位元組為單位) 太大。如需詳細資訊，請參閱Amazon Rekognition 自訂標籤中的準則。資訊清單檔案可以有少於 JSON 行的最大數目，但仍超過檔案大小上限。

您無法使用 Amazon Rekognition 自訂標籤主控台修正錯誤資訊清單檔案大小超過支援的大小上限。

修復錯誤資訊檔案大小超過支援的大小上限。

1. 檢查哪些訓練和測試資訊清單超出檔案大小上限。
2. 減少資訊清單檔案中過大的 JSON 行數。如需詳細資訊，請參閱建立清單檔案。

S3 儲存貯體許可不正確。

Amazon Rekognition 自訂標籤沒有一或多個包含訓練和測試資訊清單檔案的儲存貯體的許可。

您無法使用 Amazon Rekognition 自訂標籤主控台修正此錯誤。

若要修正錯誤 S3 儲存貯體許可不正確。

- 檢查包含訓練和測試資訊清單之值區的權限。如需詳細資訊，請參閱步驟 2：設定 Amazon Rekognition 自訂標籤主控台權限。

無法寫入至輸出 S3 儲存貯體。

服務無法產生訓練輸出檔案。

要修復錯誤無法寫入輸出 S3 存儲桶。

- 檢查OutputConfig輸入參數中的 Amazon S3 儲存貯體資訊CreateProjectVersion是否正確。

您無法使用 Amazon Rekognition 自訂標籤主控台修正此錯誤。

終端資訊內容錯誤

本主題說明資訊清單摘要中終端資訊內容錯誤報告的內容。資訊清單摘要包含每個偵測到的錯誤的錯誤碼和訊息。如需詳細資訊，請參閱了解資訊清單摘要。終端清單內容錯誤不會停止報告非終端 JSON 行驗證錯誤.

錯誤 _ 無效 _ 列資訊清單

錯誤訊息

資訊清單檔案包含太多無效資料列。

其他資訊

如ERROR_TOO_MANY_INVALID_ROWS_IN_MANIFEST果有太多 JSON 行包含無效內容，就會發生錯誤。

您無法使用 Amazon Rekognition 自訂標籤主控台修正錯誤。ERROR_TOO_MANY_INVALID_ROWS_IN_MANIFEST

若要修復錯誤 _ 多個 _ 無效 _ 列 _ 入資訊清單

1. 檢查清單是否有 JSON 行錯誤。如需詳細資訊，請參閱了解培訓和測試驗證結果清單。
2. 修正發生錯誤的 JSON 行如需詳細資訊，請參閱非終端 JSON 行驗證錯誤。

錯誤 _ 圖像 _ 在多個 _S3_ 桶

錯誤訊息

資訊清單檔案包含來自多個 S3 儲存貯體的映像。

其他資訊

資訊清單只能參考儲存在單一儲存貯體中的映像檔。每個 JSON 行都會以的值存放影像位置的 Amazon S3 位置source-ref。在下列範例中，值區名稱是我的值區。

```
"source-ref": "s3://my-bucket/images/sunrise.png"
```

您無法使用 Amazon Rekognition 自訂標籤主控台修正此錯誤。

若要修復 **ERROR_IMAGES_IN_MULTIPLE_S3_BUCKETS**

- 確保所有映像都位於同一個 Amazon S3 儲存貯體中，且每個 JSON 行`source-ref`中的值都參考了存放映像檔的儲存貯體。或者，選擇偏好的 Amazon S3 儲存貯體，然後移除`source-ref`未參考您偏好儲存貯體的 JSON 行。

錯誤 _ 無效權限 _ 映像 _ S3_ 桶

錯誤訊息

映像檔 S3 儲存貯體的權限無效。

其他資訊

包含映像檔的 Amazon S3 儲存貯體上的許可不正確。

您無法使用 Amazon Rekognition 自訂標籤主控台修正此錯誤。

若要修復 **ERROR_INVALID_PERMISSIONS_IMAGES_S3_BUCKET**

- 檢查包含映像檔的值區的權限。映像`source-ref`的值包含值區位置。

錯誤 _ 無效 _ 映像 _ S3_ 儲存格 _ 擁有者

錯誤訊息

映像 S3 儲存貯體的擁有者識別碼無效。

其他資訊

包含訓練或測試映像檔的值區擁有者與包含訓練或測試資訊清單的值區擁有者不同。您可以使用以下命令來查找值區的擁有者。

```
aws s3api get-bucket-acl --bucket bucket name
```

儲存映像檔和資訊清單檔案的值區OWNERID必須相符。

若要修復錯誤 _ 無效 _ 映像 _ S3_ 儲存格 _ 擁有者

1. 選擇所需的訓練、測試、輸出和影像儲存貯體的擁有者。擁有者必須具有使用 Amazon Rekognition 自訂標籤的許可。
2. 針對目前並非所需擁有者擁有的每個儲存貯體，建立由慣用擁有者擁有的新 Amazon S3 儲存貯體。
3. 將舊值區內容複製到新值區。如需詳細資訊，請參閱如何在 Amazon S3 儲存貯體之間複製物件？。

您無法使用 Amazon Rekognition 自訂標籤主控台修正此錯誤。

錯誤 _ 不足 _ 影像自動分割 _ 標籤 _

錯誤訊息

資訊清單檔案每個標籤包含不足的標籤影像，無法執行自動分割。

其他資訊

在模型訓練期間，您可以使用訓練資料集中 20% 的影像來建立測試資料集。當沒有足夠的影像無法建立可接受的測試資料集時，就會發生錯誤 _ 不足影像 _ PER_LABEL_FOR_自動分割。

您無法使用 Amazon Rekognition 自訂標籤主控台修正此錯誤。

修正錯誤 _ 不足 _ 影像 _ 屬性 _ 標籤 _ 自動分割

- 將更多已標記的影像新增至訓練資料集。您可以在 Amazon Rekognition 自訂標籤主控台中新增影像，方法是將映像新增至訓練資料集，或將 JSON 行新增至訓練資訊清單。如需詳細資訊，請參閱管理資料集。

錯誤清單工具標籤

錯誤訊息

資訊清單檔案的標籤太少。

其他資訊

訓練和測試資料集具有標籤所需的最低數量。最小值取決於資料集是否訓練/測試模型以偵測影像層級標籤 (分類)，或者模型是否偵測到物件位置。如果分割訓練資料集以建立測試資料集，則會在訓練資料集分割後決定資料集中的標籤數量。如需詳細資訊，請參閱Amazon Rekognition 自訂標籤中的準則。

要修復錯誤清單 _ 標籤 (控制台)

1. 在資料集中新增更多標籤。如需詳細資訊，請參閱管理標籤。
2. 將新標籤新增至資料集中的影像。如果您的型號偵測到影像層級標籤，請參閱將影像層級標籤指派給影像。如果您的模型偵測到物件位置，請參閱the section called “使用週框方塊標記物件”。

要修復錯誤資訊清單列表標籤（JSON 行）

- 為具有新標籤的新圖像添加 JSON 行。如需詳細資訊，請參閱建立清單檔案。如果您的模型偵測到影像層級標籤，您可以在class-name欄位中加入新的標籤名稱。例如，下列影像的標籤為「日出」。

```
{
    "source-ref": "s3://bucket/images/sunrise.png",
    "testdataset-classification_Sunrise": 1,
    "testdataset-classification_Sunrise-metadata": {
        "confidence": 1,
        "job-name": "labeling-job/testdataset-classification_Sunrise",
        "class-name": "Sunrise",
        "human-annotated": "yes",
        "creation-date": "2018-10-18T22:18:13.527256",
        "type": "groundtruth/image-classification"
    }
}
```

如下列範例所示，如下列範例所示。class-map

```
{
 "source-ref": "s3://custom-labels-bucket/images/IMG_1186.png",
 "bounding-box": {
  "image_size": [{
   "width": 640,
   "height": 480,
   "depth": 3
```

```
  }],
  "annotations": [{
   "class_id": 1,
   "top": 251,
   "left": 399,
   "width": 155,
   "height": 101
  }, {
   "class_id": 0,
   "top": 65,
   "left": 86,
   "width": 220,
   "height": 334
  }]
 },
 "bounding-box-metadata": {
  "objects": [{
   "confidence": 1
  }, {
   "confidence": 1
  }],
  "class-map": {
   "0": "Echo",
   "1": "Echo Dot"
  },
  "type": "groundtruth/object-detection",
  "human-annotated": "yes",
  "creation-date": "2018-10-18T22:18:13.527256",
  "job-name": "my job"
 }
}
```

您需要將類映射表映射到邊界框註釋。如需詳細資訊，請參閱清單檔案中的物件本地化。

錯誤清單多個標籤

錯誤訊息

資訊清單檔案有太多的標籤。

其他資訊

資訊清單 (資料集) 中的唯一標籤數目超過允許的限制。如果分割訓練資料集以建立測試資料集，則會在分割後決定標籤的數量。

要修復錯誤清單多標籤（控制台）

- 從資料集中移除標籤。如需詳細資訊，請參閱管理標籤。標籤會自動從資料集中的影像和邊界方框中移除。

要修復錯誤清單多個標籤（JSON 行）

- 具有圖像級 JSON 行的清單 — 如果圖像具有單個標籤，請刪除使用所需標籤的圖像的 JSON 行。如果 JSON 行包含多個標籤，請僅移除所需標籤的 JSON 物件。如需詳細資訊，請參閱對影像新增多個影像層級標籤。

  具有物件位置 JSON 行的資訊清單 — 移除要移除之標籤的邊界方框和相關標籤資訊。對包含所需標籤的每個 JSON 行執行此操作。您需要從`class-map`陣列和和陣列中移除對應物件的`objects`標籤。`annotations`如需詳細資訊，請參閱清單檔案中的物件本地化。

錯誤不足 _ 標籤 _ 重疊

錯誤訊息

訓練和測試資訊清單檔案之間的標籤重疊小於 {}%。

其他資訊

測試資料集標籤名稱和訓練資料集標籤名稱之間的重疊不到 50%。

修復錯誤不足 _ 標籤 _ 重疊（控制台）

- 從訓練資料集移除標籤。或者，您也可以在測試資料集中新增更多常用標籤。如需詳細資訊，請參閱管理標籤。標籤會自動從資料集中的影像和邊界方框中移除。

若要從訓練資料集移除標籤 (JSON 行) 來修正錯誤不足 _ 標籤 _ 重疊

- 具有圖像級 JSON 行的清單 — 如果圖像具有單個標籤，請刪除使用所需標籤的圖像的 JSON 行。如果 JSON 行包含多個標籤，請僅移除所需標籤的 JSON 物件。如需詳細資訊，請參閱對影像新增多個影像層級標籤。針對包含要移除之標籤的資訊清單中的每個 JSON 行執行此動作。

  具有物件位置 JSON 行的資訊清單 — 移除要移除之標籤的邊界方框和相關標籤資訊。對包含所需標籤的每個 JSON 行執行此操作。您需要從`class-map`陣列和和陣列中移除對應物件的`objects`標籤。`annotations`如需詳細資訊，請參閱清單檔案中的物件本地化。

通過將通用標籤添加到測試數據集（JSON 行）來修復錯誤不足 _ 標籤 _ 重疊

- 將 JSON 行新增至測試資料集，其中包含標有訓練資料集中已有標籤的影像。如需詳細資訊，請參閱建立清單檔案。

錯誤資訊清單太多可用 _ 標籤

錯誤訊息

資訊清單檔案的可用標籤太少。

其他資訊

訓練資訊清單可以包含影像層級標籤格式和物件位置格式的 JSON 行。根據訓練資訊清單中找到的 JSON 行類型，Amazon Rekognition 自訂標籤選擇建立可偵測影像層級標籤的模型，或是偵測物件位置的模型。Amazon Rekognition 自訂標籤會針對非所選格式的 JSON 行篩選出有效的 JSON 記錄。當所選模型類型資訊清單中的標籤數目不足以訓練模型時，就會發生錯誤 _ 資訊清單 _TOO_FEW_USABLE_LABLE_ 標籤。

訓練偵測影像層級標籤的型號至少需要 1 個標籤。訓練物件位置的模型至少需要 2 個標籤。

要修復錯誤清單 _ 太 _ 可用 _ 標籤 (控制台)

1. 檢查資訊清單摘要中的`use_case`欄位。
2. 針對符合的值的使用案例 (影像層級或物件當地語系化)，將更多標籤新增至訓練資料集`use_case`。如需詳細資訊，請參閱管理標籤。標籤會自動從資料集中的影像和邊界方框中移除。

要修復錯誤清單 _ 太 _ 可用 _ 標籤 (JSON 行)

1. 檢查資訊清單摘要中的use_case欄位。
2. 針對符合的值的使用案例 (影像層級或物件當地語系化)，將更多標籤新增至訓練資料集use_case。如需詳細資訊，請參閱建立清單檔案。

錯誤 _ 不足 _ 可用 _ 標籤 _ 重疊

錯誤訊息

訓練和測試資訊清單檔案之間的可用標籤重疊小於 {}%。

其他資訊

訓練資訊清單可以包含影像層級標籤格式和物件位置格式的 JSON 行。根據訓練資訊清單中找到的格式，Amazon Rekognition 自訂標籤選擇建立可偵測影像層級標籤的模型，或是偵測物件位置的模型。Amazon Rekognition 自訂標籤不會針對非所選模型格式的 JSON 行使用有效的 JSON 記錄。當所使用的測試與訓練標籤之間的重疊小於 50% 時，就會發生錯誤 _ 不足 _ 可用性 _ 標籤 _ 重疊。

修復錯誤不足 _ 可用 _ 標籤 _ 重疊（控制台）

- 從訓練資料集移除標籤。或者，您也可以在測試資料集中新增更多常用標籤。如需詳細資訊，請參閱管理標籤。標籤會自動從資料集中的影像和邊界方框中移除。

若要從訓練資料集移除標籤 (JSON 行) 來修正錯誤不足 _ 可用性 _ 標籤 _ 重疊

- 用於偵測影像層級標籤的資料集 — 如果影像具有單一標籤，請移除使用所需標籤之影像的 JSON Line。如果 JSON 行包含多個標籤，請僅移除所需標籤的 JSON 物件。如需詳細資訊，請參閱對影像新增多個影像層級標籤。針對包含要移除之標籤的資訊清單中的每個 JSON 行執行此動作。

  用來偵測物件位置的資料集 — 移除要移除之標籤的邊界方框和相關標籤資訊。對包含所需標籤的每個 JSON 行執行此操作。您需要從class-map陣列和和陣列中移除對應物件的objects標籤。annotations如需詳細資訊，請參閱清單檔案中的物件本地化。

通過向測試數據集添加通用標籤來修復錯誤不足 _ 可用 _ 標籤 _ 重疊（JSON 行）

- 將 JSON 行新增至測試資料集，其中包含標有訓練資料集中已有標籤的影像。如需詳細資訊，請參閱建立清單檔案。

錯誤 _ 失敗 _ 影像 _S3_ 複製

錯誤訊息

無法從 S3 儲存貯體複製映像。

其他資訊

服務無法複製資料集中的任何影像。

您無法使用 Amazon Rekognition 自訂標籤主控台修正此錯誤。

修復錯誤 _ 失敗 _ 影像 _S3_ 複製

1. 檢查圖像的權限。
2. 如果您正在使用AWS KMS，請檢查值區政策。如需詳細資訊，請參閱解密使用 AWS Key Management Service 加密的文件。

資訊清單檔案有太多的終端機錯誤。

終端機內容錯誤的 JSON 行太多。

若要修復 **ERROR_TOO_MANY_RECORDS_IN_ERROR**

- 減少終端機內容錯誤的 JSON 行（圖像）的數量。如需詳細資訊，請參閱終端資訊內容錯誤。

您無法使用 Amazon Rekognition 自訂標籤主控台修正此錯誤。

非終端 JSON 行驗證錯誤

本主題列出 Amazon Rekognition 自訂標籤在訓練期間報告的非終端 JSON 行驗證錯誤。這些錯誤會在訓練和測試驗證資訊清單中報告。如需詳細資訊，請參閱了解培訓和測試驗證結果清單。您可以透過更新訓練或測試資訊清單檔案中的 JSON 行來修正非終端 JSON 行錯誤。您也可以從資訊清單中移除 JSON Line，但這樣做可能會降低模型的品質。如果有許多非終端驗證錯誤，您可能會發現重新創建清

單文件更容易。驗證錯誤通常發生在手動創建的清單文件中。如需詳細資訊，請參閱建立清單檔案。如需修正驗證錯誤的資訊，請參閱修正訓練錯誤。您可以使用 Amazon Rekognition 自訂標籤主控台來修正某些錯誤。

錯誤 _ 遺失 _ 來源 _ 參考

錯誤訊息

來源參考索引鍵遺失。

其他資訊

JSON 行`source-ref`欄位提供影像的 Amazon S3 位置。遺失`source-ref`金鑰或拼寫錯誤時，會發生此錯誤。此錯誤通常發生在手動創建的清單文件中。如需詳細資訊，請參閱建立清單檔案。

若要修復 **ERROR_MISSING_SOURCE_REF**

1. 檢查`source-ref`密鑰是否存在並且拼寫正確。完整的`source-ref`鍵和值類似於以下內容。是`"source-ref": "s3://bucket/path/image"`。
2. 更新或 JSON 行中的`source-ref`密鑰。或者，從資訊清單檔案中移除 JSON 行。

您無法使用 Amazon Rekognition 自訂標籤主控台修正此錯誤。

錯誤 _ 無效 _ 來源 _ 參考格式

錯誤訊息

來源參考值的格式無效。

其他資訊

`source-ref`金鑰存在於 JSON 行中，但 Amazon S3 路徑的結構描述不正確。例如，路徑`https://....`代替`S3://....`。在手動建立的資訊清單檔案中，通常會發生錯誤 _ 無效 _ 來源 _ 參考 _ 格式錯誤。如需詳細資訊，請參閱建立清單檔案。

若要修復 **ERROR_INVALID_SOURCE_REF_FORMAT**

1. 檢查結構描述是否為`"source-ref": "s3://bucket/path/image"`。例如：`"source-ref": "s3://custom-labels-console-us-east-1-1111111111/images/000000242287.jpg"`。
2. 更新或移除資訊清單檔案中的 JSON 行。

您無法使用 Amazon Rekognition 自訂標籤主控台來修正此問題。ERROR_INVALID_SOURCE_REF_FORMAT

錯誤標籤屬性

錯誤訊息

找不到標籤屬性。

其他資訊

label 屬性或 label 屬性-metadata鍵名稱（或兩者）無效或缺少。在下列範例中，只要遺失bounding-box或bounding-box-metadata金鑰 (或兩者)，就ERROR_NO_LABEL_ATTRIBUTES會發生。如需詳細資訊，請參閱建立清單檔案。

```
{
 "source-ref": "s3://custom-labels-bucket/images/IMG_1186.png",
 "bounding-box": {
  "image_size": [{
   "width": 640,
   "height": 480,
   "depth": 3
  }],
  "annotations": [{
   "class_id": 1,
   "top": 251,
   "left": 399,
   "width": 155,
   "height": 101
  }, {
   "class_id": 0,
   "top": 65,
   "left": 86,
   "width": 220,
   "height": 334
  }]
 },
 "bounding-box-metadata": {
  "objects": [{
   "confidence": 1
  }, {
   "confidence": 1
  }],
  "class-map": {
```

```
    "0": "Echo",
    "1": "Echo Dot"
   },
   "type": "groundtruth/object-detection",
   "human-annotated": "yes",
   "creation-date": "2018-10-18T22:18:13.527256",
   "job-name": "my job"
  }
}
```

ERROR_NO_LABEL_ATTRIBUTES錯誤通常發生在手動創建的清單文件中。如需詳細資訊，請參閱建立清單檔案。

若要修復 **ERROR_NO_LABEL_ATTRIBUTES**

1. 檢查 label 屬性標識符和 label 屬性標識符標識-metadata鍵是否存在，以及密鑰名稱拼寫正確。
2. 更新或移除資訊清單檔案中的 JSON 行。

您無法使用 Amazon Rekognition 自訂標籤主控台修正。ERROR_NO_LABEL_ATTRIBUTES

錯誤 _ 無效 _ 標籤屬性 _ 格式

錯誤訊息

標籤屬性 {} 的格式無效。

其他資訊

標籤屬性索引鍵的結構描述遺失或無效。手動建立的資訊清單檔案通常會發生錯誤。如需詳細資訊，請參閱。建立清單檔案

若要修復 **ERROR_INVALID_LABEL_ATTRIBUTE_FORMAT**

1. 檢查標籤屬性鍵的 JSON 行部分是否正確。在下列範例物件位置範例中，image_size和annotations物件必須正確。標籤屬性鍵被命名bounding-box。

```
"bounding-box": {
 "image_size": [{
  "width": 640,
  "height": 480,
  "depth": 3
 }],
```

```
 "annotations": [{
  "class_id": 1,
  "top": 251,
  "left": 399,
  "width": 155,
  "height": 101
 }, {
  "class_id": 0,
  "top": 65,
  "left": 86,
  "width": 220,
  "height": 334
 }]
},
```

2. 更新或移除資訊清單檔案中的 JSON 行。

您無法使用 Amazon Rekognition 自訂標籤主控台修正此錯誤。

錯誤 _ 無效 _ 標籤屬性 _ 中繼資料格式

錯誤訊息

標籤屬性中繼資料的格式無效。

其他資訊

標籤屬性中繼資料索引鍵的結構描述遺失或無效。通常在手動創建的清單文件中發生錯誤 _ 無效 _ 標籤屬性 _ 元數據庫 _ 格式錯誤。如需詳細資訊，請參閱建立清單檔案。

若要修復 `ERROR_INVALID_LABEL_ATTRIBUTE_FORMAT`

1. 檢查標籤屬性中繼資料索引鍵的 JSON Line 結構描述是否類似於下列範例。label 屬性中繼資料索引鍵已命名為`bounding-box-metadata`。

```
"bounding-box-metadata": {
 "objects": [{
  "confidence": 1
 }, {
  "confidence": 1
 }],
 "class-map": {
```

```
   "0": "Echo",
   "1": "Echo Dot"
  },
  "type": "groundtruth/object-detection",
  "human-annotated": "yes",
  "creation-date": "2018-10-18T22:18:13.527256",
  "job-name": "my job"
 }
```

2. 更新或移除資訊清單檔案中的 JSON 行。

您無法使用 Amazon Rekognition 自訂標籤主控台修正此錯誤。

有效的標籤屬性錯誤

錯誤訊息

找不到有效的標籤屬性。

其他資訊

在 JSON 行中找不到有效的標籤屬性。Amazon Rekognition 自訂標籤會同時檢查標籤屬性和標籤屬性識別碼。手動建立的資訊清單檔案通常會發生錯誤。如需詳細資訊，請參閱。建立清單檔案

如果 JSON 行不是支援的SageMaker資訊清單格式，Amazon Rekognition 自訂標籤會將 JSON 行標記為無效，並報告錯ERROR_NO_VALID_LABEL_ATTRIBUTES誤。目前 Amazon Rekognition 自訂標籤支援分類任務和邊界框格式。如需詳細資訊，請參閱建立清單檔案。

若要修復 **ERROR_NO_VALID_LABEL_ATTRIBUTES**

1. 檢查標籤屬性金鑰和標籤屬性中繼資料的 JSON 是否正確。
2. 更新或移除資訊清單檔案中的 JSON 行。如需詳細資訊，請參閱the section called “建立清單檔案”。

您無法使用 Amazon Rekognition 自訂標籤主控台修正此錯誤。

錯誤 _ 遺失 _ 邊界 _ 信心

錯誤訊息

一或多個邊界方框缺少置信度值。

其他資訊

一或多個物件位置邊界方框的信賴鍵遺失。邊界方框的信賴度索引鍵位於 label 屬性中繼資料夾，如下列範例所示。錯誤 _ 遺失 _ 邊界 _ 置信度錯誤通常發生在手動建立的資訊清單檔案。如需詳細資訊，請參閱the section called “清單檔案中的物件本地化”。

```
"bounding-box-metadata": {
 "objects": [{
  "confidence": 1
 }, {
  "confidence": 1
 }],
```

若要修復 **ERROR_MISSING_BOUNDING_BOX_CONFIDENCE**

1. 檢查 label 屬性中的`objects`陣列是否包含相同數量的置信鍵，因為有在 label 屬性`annotations`陣列中的對象。
2. 更新或移除資訊清單檔案中的 JSON 行。

您無法使用 Amazon Rekognition 自訂標籤主控台修正此錯誤。

錯誤 _ 缺少類別 _ 地圖識別碼

錯誤訊息

類別對應中缺少其中一個類別 ID。

其他資訊

`class_id`在註釋中（邊界框）對象在標籤屬性元數據類 map（`class-map`）中沒有匹配條目。如需詳細資訊，請參閱清單檔案中的物件本地化。通常在手動建立的資訊清單檔案中發生錯誤。

若要修正錯誤 _ 遺失 _ 類別 _ 對應 ID

1. 檢查每個註釋 (邊界框) 物件中的值是否在`class-map`陣列中具有對應的值，如下列範例所示。`class_id`數`annotations`組和數`class_map`組應具有相同數量的元素。

```
{
 "source-ref": "s3://custom-labels-bucket/images/IMG_1186.png",
```

```
 "bounding-box": {
  "image_size": [{
   "width": 640,
   "height": 480,
   "depth": 3
  }],
  "annotations": [{
   "class_id": 1,
   "top": 251,
   "left": 399,
   "width": 155,
   "height": 101
  }, {
   "class_id": 0,
   "top": 65,
   "left": 86,
   "width": 220,
   "height": 334
  }]
 },
 "bounding-box-metadata": {
  "objects": [{
   "confidence": 1
  }, {
   "confidence": 1
  }],
  "class-map": {
   "0": "Echo",
   "1": "Echo Dot"
  },
  "type": "groundtruth/object-detection",
  "human-annotated": "yes",
  "creation-date": "2018-10-18T22:18:13.527256",
  "job-name": "my job"
 }
}
```

2. 更新或移除資訊清單檔案中的 JSON 行。

您無法使用 Amazon Rekognition 自訂標籤主控台修正此錯誤。

錯誤 _ 無效 _ J SON_ 行

錯誤訊息

JSON 行的格式無效。

其他資訊

在 JSON 行中發現未預期的字元。JSON 行會取代為只包含錯誤資訊的新 JSON 行。錯誤 _ 無效 _ JSON_LINE 錯誤通常發生在手動建立的資訊清單檔案中。如需詳細資訊，請參閱the section called “清單檔案中的物件本地化”。

您無法使用 Amazon Rekognition 自訂標籤主控台修正此錯誤。

若要修復 **ERROR_INVALID_JSON_LINE**

1. 開啟資訊清單檔案，並導覽至發生錯誤 _ 無效 _JSON_LINE 錯誤的 JSON 行。
2. 檢查 JSON 行不包含無效字元，且必要,字元;或字元未遺失。
3. 更新或移除資訊清單檔案中的 JSON 行。

錯誤 _ 無效影像

錯誤訊息

影像無效。檢查 S3 路徑和/或映像屬性。

其他資訊

參考的檔案不`source-ref`是有效的影像。可能的原因包括圖像縱橫比，圖像的大小和圖像格式。

如需詳細資訊，請參閱準則。

若要修復 **ERROR_INVALID_IMAGE**

1. 請檢查以下內容。
   - 圖像的縱橫比小於 20：1。
   - 圖像的大小大於 15 MB
   - 影像為 PNG 或 JPEG 格式。
   - 中的影像路徑`source-ref`正確。
   - 影像的最小影像尺寸大於 64 像素 x 64 像素。

- 影像的最大影像尺寸小於 4096 像素 x 4096 像素。

2. 更新或移除資訊清單檔案中的 JSON 行。

您無法使用 Amazon Rekognition 自訂標籤主控台修正此錯誤。

錯誤 _ 無效影像尺寸

錯誤訊息

影像尺寸不符合允許的尺寸。

其他資訊

參照的影像`source-ref`不符合允許的影像尺寸。最小尺寸為 64 像素。最大尺寸為 4096 像素。`ERROR_INVALID_IMAGE_DIMENSION`報告含有邊界方框的影像。

如需詳細資訊，請參閱準則。

要修復**`ERROR_INVALID_IMAGE_DIMENSION`**（控制台）

1. 使用 Amazon Rekognition 自訂標籤可處理的維度來更新 Amazon S3 儲存貯體中的映像。
2. 在 Amazon Rekognition 自訂標籤主控台中，請執行以下操作：

   a. 從影像中移除現有的邊界方框。

   b. 將邊界方框重新增至影像。

   c. 儲存您的變更。

   如需詳細資訊，請參閱「使用週框方塊標記物件」。

若要修正問題 **`ERROR_INVALID_IMAGE_DIMENSION`** (SDK)

1. 使用 Amazon Rekognition 自訂標籤可處理的維度來更新 Amazon S3 儲存貯體中的映像。
2. 通過調用獲取圖像的現有 JSON 行ListDatasetEntries。對於`SourceRefContains`輸入參數，指定映像檔的 Amazon S3 位置和檔案名稱。
3. 呼叫UpdateDatasetEntries並提供影像的 JSON 行。請確定的值`source-ref`符合 Amazon S3 儲存貯體中的映像位置。更新邊界框註釋，以符合更新影像所需的邊界框尺寸。

```
{
```

```
 "source-ref": "s3://custom-labels-bucket/images/IMG_1186.png",
 "bounding-box": {
  "image_size": [{
   "width": 640,
   "height": 480,
   "depth": 3
  }],
  "annotations": [{
   "class_id": 1,
   "top": 251,
   "left": 399,
   "width": 155,
   "height": 101
  }, {
   "class_id": 0,
   "top": 65,
   "left": 86,
   "width": 220,
   "height": 334
  }]
 },
 "bounding-box-metadata": {
  "objects": [{
   "confidence": 1
  }, {
   "confidence": 1
  }],
  "class-map": {
   "0": "Echo",
   "1": "Echo Dot"
  },
  "type": "groundtruth/object-detection",
  "human-annotated": "yes",
  "creation-date": "2013-11-18T02:53:27",
  "job-name": "my job"
 }
}
```

錯誤 _ 無效 _ 邊界 _ 方塊

錯誤訊息

邊界方框具有關閉框架值。

其他資訊

邊界框資訊會指定影像框外或包含負值的影像。

如需詳細資訊，請參閱準則。

若要修復 ERROR_INVALID_BOUNDING_BOX

1. 檢查陣列中邊界方塊的annotations值。

```
"bounding-box": {
 "image_size": [{
  "width": 640,
  "height": 480,
  "depth": 3
 }],
 "annotations": [{
  "class_id": 1,
  "top": 251,
  "left": 399,
  "width": 155,
  "height": 101
 }]
},
```

2. 從資訊清單檔案更新或移除 JSON 行。

您無法使用 Amazon Rekognition 自訂標籤主控台修正此錯誤。

錯誤有效註釋

錯誤訊息

找不到有效的註釋。

其他資訊

JSON 行中沒有任何註釋物件包含有效的邊界框資訊。

若要修復 ERROR_NO_VALID_ANNOTATIONS

1. 更新annotations陣列以包含有效的邊界框物件。此外，請檢查標籤屬性中繼資料中對應的邊界方框資訊 (confidence和class_map) 是否正確。如需詳細資訊，請參閱清單檔案中的物件本地化。

```
{
 "source-ref": "s3://custom-labels-bucket/images/IMG_1186.png",
 "bounding-box": {
  "image_size": [{
   "width": 640,
   "height": 480,
   "depth": 3
  }],
  "annotations": [
     {
   "class_id": 1,    #annotation object
   "top": 251,
   "left": 399,
   "width": 155,
   "height": 101
  }, {
   "class_id": 0,
   "top": 65,
   "left": 86,
   "width": 220,
   "height": 334
  }]
 },
 "bounding-box-metadata": {
  "objects": [
  >{
   "confidence": 1          #confidence  object
  },
        {
   "confidence": 1
  }],
  "class-map": {
   "0": "Echo",    #label
   "1": "Echo Dot"
  },
  "type": "groundtruth/object-detection",
  "human-annotated": "yes",
```

```
  "creation-date": "2018-10-18T22:18:13.527256",
  "job-name": "my job"
 }
}
```

2. 從資訊清單檔案更新或移除 JSON 行。

您無法使用 Amazon Rekognition 自訂標籤主控台修正此錯誤。

錯誤 _ 邊界 _ 方塊 _ 太小

錯誤訊息

邊界方框的高度和寬度太小。

其他資訊

邊界方框尺寸 (高度和寬度) 必須大於 1 x 1 像素。

在訓練期間，如果影像的任何尺寸大於 1280 像素 (來源映像不會受到影響)，Amazon Rekognition 自訂標籤會調整影像的大小。產生的邊界方框高度和寬度必須大於 1 x 1 像素。邊界框位置存儲在對象位置 JSON 行的`annotations`數組中。如需詳細資訊，請參閱 清單檔案中的物件本地化

```
"bounding-box": {
 "image_size": [{
  "width": 640,
  "height": 480,
  "depth": 3
 }],
 "annotations": [{
  "class_id": 1,
  "top": 251,
  "left": 399,
  "width": 155,
  "height": 101
 }]
},
```

會將錯誤資訊加入至註解物件。

要修復錯誤 _ 邊界 _ 箱 _ 太小

- 選擇下列其中一個選項。
  - 增加太小的邊界方框的大小。
  - 移除太小的邊界方框。如需有關移除邊界框的資訊，請參閱錯誤 _ 多個 _ 彈出 _ 方塊。
  - 從清單中刪除圖像（JSON 行）。

錯誤 _ 多個 _ 彈出 _ 方塊

錯誤訊息

邊界方框超過允許的最大值。

其他資訊

邊界方框數量超過允許的限制 (50)。您可以在 Amazon Rekognition 自訂標籤主控台中移除多餘的邊界方塊，也可以從 JSON 行中移除它們。

要修復**ERROR_TOO_MANY_BOUNDING_BOXES**（控制台）。

1. 決定要移除的邊界方框。
2. 開啟亞馬遜重新認知主控台，網址為 https://console.aws.amazon.com/rekognition/。
3. 選擇「使用自訂標籤」。
4. 選擇 Get started (開始使用)。
5. 在左側導覽窗格中，請選擇包含您要使用之資料集的專案。
6. 在 [資料集] 區段中，請選擇您要使用的資料集。
7. 在資料集收藏館頁面中，選擇 [開始標記] 以進入標籤模式。
8. 選擇您要從中移除邊界方框的影像。
9. 選擇「繪製邊界方框」。
10. 在繪圖工具中，請選擇您要刪除的邊界方框。
11. 按下鍵盤上的 delete 鍵可刪除邊界方框。
12. 重複前兩個步驟，直到刪除足夠的邊界方框為止。

13. 選擇完成
14. 請選擇 Save changes (儲存變更) 儲存您所做的變更。
15. 選擇 [結束] 以結束標籤模式。

要修復錯誤 _ 多個 _ 彈出框（JSON 行）。

1. 開啟資訊清單檔案，並導覽至 JSON 行，其中發生錯誤。
2. 請為您要移除的每個邊界方框移除以下內容。

   - 從數組中刪除所需的annotation對annotations象。
   - 從 label 屬性元數據中的objects數組中刪除相應的對confidence象。
   - 如果其他邊界方框不再使用，請從中移除標籤class-map。

   使用下列範例來識別要移除的項目。

```
{
 "source-ref": "s3://custom-labels-bucket/images/IMG_1186.png",
 "bounding-box": {
  "image_size": [{
   "width": 640,
   "height": 480,
   "depth": 3
  }],
  "annotations": [
     {
   "class_id": 1,    #annotation object
   "top": 251,
   "left": 399,
   "width": 155,
   "height": 101
  }, {
   "class_id": 0,
   "top": 65,
   "left": 86,
   "width": 220,
   "height": 334
  }]
 },
 "bounding-box-metadata": {
```

```
  "objects": [
  >{
   "confidence": 1          #confidence  object
  },
        {
   "confidence": 1
  }],
  "class-map": {
   "0": "Echo",    #label
   "1": "Echo Dot"
  },
  "type": "groundtruth/object-detection",
  "human-annotated": "yes",
  "creation-date": "2018-10-18T22:18:13.527256",
  "job-name": "my job"
 }
}
```

警告 _ 未註解 _ 記錄

警告訊息

記錄不加註釋。

其他資訊

使用 Amazon Rekognition 自訂標籤主控台新增至資料集的映像並未加上標籤。圖像的 JSON 行不用於訓練。

```
{
    "source-ref": "s3://bucket/images/IMG_1186.png",
    "warnings": [
        {
            "code": "WARNING_UNANNOTATED_RECORD",
            "message": "Record is unannotated."
        }
    ]
}
```

修正警告 _ 未註釋 _ 記錄

- 使用亞馬遜自訂標籤主控台為映像加上標籤。如需相關指示，請參閱將影像層級標籤指派給影像。

警告無註解

警告訊息

未提供任何註解。

其他資訊

物件當地語系化格式的 JSON 行不包含任何邊界方塊資訊，儘管有人 (human-annotated = yes) 加上註解。JSON 行是有效的，但不用於培訓。如需詳細資訊，請參閱了解培訓和測試驗證結果清單。

```
{
    "source-ref": "s3://bucket/images/IMG_1186.png",
    "bounding-box": {
        "image_size": [
            {
                "width": 640,
                "height": 480,
                "depth": 3
            }
        ],
        "annotations": [

        ],
        "warnings": [
            {
                "code": "WARNING_NO_ATTRIBUTE_ANNOTATIONS",
                "message": "No attribute annotations were found."
            }
        ]
    },
    "bounding-box-metadata": {
        "objects": [

        ],
        "class-map": {

        },
```

```
        "type": "groundtruth/object-detection",
        "human-annotated": "yes",
        "creation-date": "2013-11-18 02:53:27",
        "job-name": "my job"
    },
    "warnings": [
        {
            "code": "WARNING_NO_ANNOTATIONS",
            "message": "No annotations were found."
        }
    ]
}
```

欲修復警告 _ 無註釋

- 選擇下列其中一個選項。
  - 將邊界方框 (annotations) 資訊新增至 JSON 行。如需詳細資訊，請參閱清單檔案中的物件本地化。
  - 從清單中刪除圖像（JSON 行）。

警告不屬性註釋

警告訊息

未提供屬性註釋。

其他資訊

物件當地語系化格式的 JSON 行不包含任何邊界框註解資訊，儘管有人 (human-annotated = yes) 加上註解。annotations陣列不存在或未填入。JSON 行是有效的，但不用於培訓。如需詳細資訊，請參閱了解培訓和測試驗證結果清單。

```
{
    "source-ref": "s3://bucket/images/IMG_1186.png",
    "bounding-box": {
        "image_size": [
            {
                "width": 640,
                "height": 480,
```

```
                "depth": 3
            }
        ],
        "annotations": [

        ],
        "warnings": [
            {
                "code": "WARNING_NO_ATTRIBUTE_ANNOTATIONS",
                "message": "No attribute annotations were found."
            }
        ]
    },
    "bounding-box-metadata": {
        "objects": [

        ],
        "class-map": {

        },
        "type": "groundtruth/object-detection",
        "human-annotated": "yes",
        "creation-date": "2013-11-18 02:53:27",
        "job-name": "my job"
    },
    "warnings": [
        {
            "code": "WARNING_NO_ANNOTATIONS",
            "message": "No annotations were found."
        }
    ]
}
```

修復警告 _ 無 _ 屬性 _ 註釋

- 選擇下列其中一個選項。
  - 將一個或多個邊界框annotation對象添加到 JSON 行。如需詳細資訊，請參閱清單檔案中的物件本地化。
  - 移除邊界框屬性。
  - 從清單中刪除圖像（JSON 行）。如果 JSON 行中存在其他有效的邊界框屬性，您可以改為只從 JSON 行中移除無效的邊界框屬性。

不支援的 _ 使用案例類型

警告訊息

其他資訊

type 欄位的值不是groundtruth/image-classification或groundtruth/object-detection。如需詳細資訊，請參閱建立清單檔案。

```
{
    "source-ref": "s3://bucket/test_normal_8.jpg",
    "BB": {
        "annotations": [
            {
                "left": 1768,
                "top": 1007,
                "width": 448,
                "height": 295,
                "class_id": 0
            },
            {
                "left": 1794,
                "top": 1306,
                "width": 432,
                "height": 411,
                "class_id": 1
            },
            {
                "left": 2568,
                "top": 1346,
                "width": 710,
                "height": 305,
                "class_id": 2
            },
            {
                "left": 2571,
                "top": 1020,
                "width": 644,
                "height": 312,
                "class_id": 3
            }
        ],
        "image_size": [
            {
```

```
                "width": 4000,
                "height": 2667,
                "depth": 3
            }
        ]
    },
    "BB-metadata": {
        "job-name": "labeling-job/BB",
        "class-map": {
            "0": "comparator",
            "1": "pot_resistor",
            "2": "ir_phototransistor",
            "3": "ir_led"
        },
        "human-annotated": "yes",
        "objects": [
            {
                "confidence": 1
            },
            {
                "confidence": 1
            },
            {
                "confidence": 1
            },
            {
                "confidence": 1
            }
        ],
        "creation-date": "2021-06-22T09:58:34.811Z",
        "type": "groundtruth/wrongtype",
        "cl-errors": [
            {
                "code": "ERROR_UNSUPPORTED_USE_CASE_TYPE",
                "message": "The use case type of the BB-metadata label attribute
metadata is unsupported. Check the type field."
            }
        ]
    },
    "cl-metadata": {
        "is_labeled": true
    },
    "cl-errors": [
        {
```

```
            "code": "ERROR_NO_VALID_LABEL_ATTRIBUTES",
            "message": "No valid label attributes found."
        }
    ]
}
```

若要修正錯誤 _ 不支援的 _ 使用案例類型

- 請選擇下列其中一個選項：
  - 根據您要建立的模型類型`groundtruth/object-detection`，將`type`欄位的值變更為`groundtruth/image-classification`或。如需詳細資訊，請參閱建立清單檔案。
  - 從清單中刪除圖像（JSON 行）。

錯誤 _ 無效 _ 標籤名稱 _ 長度

其他資訊

標籤名稱的長度太長。長度上限為 256 個字元。

修復錯誤 _ 無效 _ 標籤 _ 名稱 _ 長度

- 請選擇下列其中一個選項：
  - 將標籤名稱的長度縮短為 256 個字元或更少。
  - 從清單中刪除圖像（JSON 行）。

改善訓練有素的 Amazon Rekognition 自訂標籤模型

訓練完成後，您可以評估模型的效能。為了協助您，Amazon Rekognition 自訂標籤提供每個標籤的摘要指標和評估指標。如需可用指標的相關資訊，請參閱評估模型的指標。若要使用量度改善模型，請參閱改善 Amazon Rekognition 自訂標籤模型。

如果對模型的準確性滿意，您還可以開始使用它。如需詳細資訊，請參閱執行培訓過的 Amazon Rekognition 自訂標籤模型。

主題

評估模型的指標

模型訓練完畢後，Amazon Rekognition 自訂標籤會從模型測試傳回指標，您可以使用這些指標來評估模型的效能。本主題說明您可以使用的量度，以及如何瞭解訓練過的模型是否成效良好。

Amazon Rekognition 自訂標籤主控台提供下列指標，做為訓練結果摘要和每個標籤的指標：

- 精確度
- 取回
- F1

我們提供的每個量度都是評估 Machine Learning 模型效能的常用量度。Amazon Rekognition 自訂標籤會傳回整個測試資料集的測試結果指標，以及每個自訂標籤的指標。您也可以針對測試資料集中每個映像檔檢閱訓練過的自訂模型效能。如需詳細資訊，請參閱存取評估指標 (主控台)。

評估模型效能

在測試期間，Amazon Rekognition 自訂標籤會預測測試影像是否包含自訂標籤。置信度分數是量化模型預測確定性的值。

如果自訂標籤的可信度分數超過閾值，則模型輸出將包含此標籤。預測可以通過以下方式分類：

- 正面 — Amazon Rekognition 自訂標籤模型可正確預測測試影像中自訂標籤的存在。也就是說，預測的標籤也是該圖像的「基本真相」標籤。例如，當影像中有足球時，Amazon Rekognition 自訂標籤會正確傳回足球標籤。
- 誤報 — Amazon Rekognition 自訂標籤模型不正確地預測測試影像中自訂標籤的存在。也就是說，預測的標籤不是圖像的基本真值標籤。例如，Amazon Rekognition 自訂標籤會傳回足球標籤，但該影像的地面真相中沒有足球標籤。
- 誤報 — Amazon Rekognition 自訂標籤模型未預測影像中存在自訂標籤，但該映像的「基本真相」包含此標籤。例如，Amazon Rekognition 自訂標籤不會針對包含足球的影像傳回「足球」自訂標籤。
- 真正的負面 — Amazon Rekognition 自訂標籤模型可正確預測測試影像中不存在自訂標籤。例如，Amazon Rekognition 自訂標籤不會針對不包含足球的影像傳回足球標籤。

控制台提供對測試數據集中每個圖像的真正值，誤報和假負值的訪問。如需詳細資訊，請參閱存取評估指標 (主控台)。

這些預測結果用於計算每個標籤的以下指標，以及整個測試集的彙總。相同的定義適用於模型在邊界框層級所做的預測，區別在於所有量度都是根據每個測試圖像中的每個邊界框（預測或基準真值）進行計算。

聯集上的交集 (IoU) 和物體偵測

「聯集上的交集」(IoU) 會測量兩個物件邊界方框在其組合區域上重疊的百分比。範圍為 0 (最低重疊) 到 1 (完全重疊)。在測試期間，當接地真值邊界方塊和預測邊界方塊的 IoU 至少為 0.5 時，預測的邊界方塊是正確的。

假設閾值

Amazon Rekognition 自訂標籤會自動計算每個自訂標籤的假設臨界值 (0-1)。您無法為自訂標籤設定假設的閾值。每個標籤的假設臨界值都是高於此值，預測會計為真正或假陽性。它是根據您的測試數據集設置的。假設的臨界值是根據模型訓練期間在測試資料集上達到的最佳 F1 分數來計算。

您可以從模型的訓練結果中取得標籤的假設臨界值。如需詳細資訊，請參閱存取評估指標 (主控台)。

對假設臨界值的變更通常用於改善模型的精確度和重新叫用。如需詳細資訊，請參閱改善 Amazon Rekognition 自訂標籤模型。由於您無法為標籤設置模型的假定閾值，因此可以通過分析圖像`DetectCustomLabels`並指定`MinConfidence`輸入參數來實現相同的結果。如需詳細資訊，請參閱使用經過培訓的模型分析圖像。

精確度

Amazon Rekognition 自訂標籤可為每個標籤提供精確度指標，以及整個測試資料集的平均精確度指標。

精確度是在個別標籤的假設閾值上，所有模型預測 (真正值和誤報) 的正確預測 (真正正值) 的分數。隨著臨界值的增加，模型可能會進行較少的預測。但是，一般而言，與較低的閾值相比，它的真正正值與誤報比例更高。精確度的可能值範圍為 0—1，而較高的值表示較高的精確度。

例如，當模型預測圖像中存在足球時，該預測是否正確？ 假設有 8 個足球和 5 個岩石的圖像。如果模型預測 9 個足球-8 個正確預測和 1 個誤報-那麼這個例子的精度是 0.89。但是，如果模型預測了圖像中的 13 個足球，8 個正確的預測和 5 個不正確，則得到的精度會較低。

如需詳細資訊，請參閱精確度和召回。

取回

Amazon Rekognition 自訂標籤提供每個標籤的平均召回指標，以及整個測試資料集的平均召回指標。

召回是正確預測高於假定閾值的測試集標籤的分數。這是一種測量模型在測試集圖像中實際存在時可以正確預測自定義標籤的頻率。召回的範圍是 0 到 1。較高的值表示召回率越高。

例如，如果一個圖像包含 8 個足球，其中有多少是正確的檢測？ 在此範例中，影像有 8 個足球和 5 個岩石，如果模型偵測到 5 個足球，則召回值為 0.62。如果在重新訓練後，新模型檢測到 9 個足球，包括圖像中存在的所有 8 個球，則召回值為 1.0。

如需詳細資訊，請參閱精確度和召回。

F1

Amazon Rekognition 自訂標籤使用 F1 評分指標來測量每個標籤的平均模型效能，以及整個測試資料集的平均模型效能。

模型效能是一種彙總度量，會考慮到所有標籤的精確度和召回。 (例如 F1 分數或平均精確度)。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。具體而言，分類工作的模型效能通常是以 F1 分數來衡量。該分數是在假定閾值處精度和召回分數的諧波平均值。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。

F1 分數的高值表示模型在精確度和調用方面都表現良好。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，並且表示模型出現問題。

如需詳細資訊，請參閱 F1 分數。

使用 指標

對於您已訓練並根據您的應用程式而定的給定模型，您可以使用MinConfidence input 參數來在精度和調用之間進行折衷DetectCustomLabels。以較高的MinConfidence價值，您通常會獲得更高的精度（足球的更正確的預測），但較低的回憶（更多的實際足球將被錯過）。在較低的MinConfidence值，你會得到更高的召回（更多的實際足球正確預測），但更低的精度（更多的這些預測將是錯誤的）。如需詳細資訊，請參閱使用經過培訓的模型分析圖像。

如果需要，這些指標也會通知您可能採取的步驟來改善模型效能。如需詳細資訊，請參閱改善 Amazon Rekognition 自訂標籤模型。

> **Note**
>
> DetectCustomLabels傳回範圍從 0 到 100 的預測值，這些預測值對應於 0-1 的度量範圍。

存取評估指標 (主控台)

在測試期間，會針對測試資料集評估模型的效能。在測試數據集的標籤被認為是「地面真實」，因為它們代表了實際的圖像代表什麼。在測試期間，模型會使用測試資料集進行預測。預測的標籤會與地面真值標籤進行比較，結果可在主控台評估頁面中取得。

Amazon Rekognition 自訂標籤主控台會顯示整個模型的摘要指標和個別標籤的指標。主控台中可用的指標包括精確回復、F1 分數、可信度和可信度閾值。如需詳細資訊，請參閱改善訓練有素的 Amazon Rekognition 自訂標籤模型。

您可以使用主控台專注於個別指標。例如，若要調查標籤的精確度問題，您可以依標籤和誤判結果篩選訓練結果。如需詳細資訊，請參閱評估模型的指標。

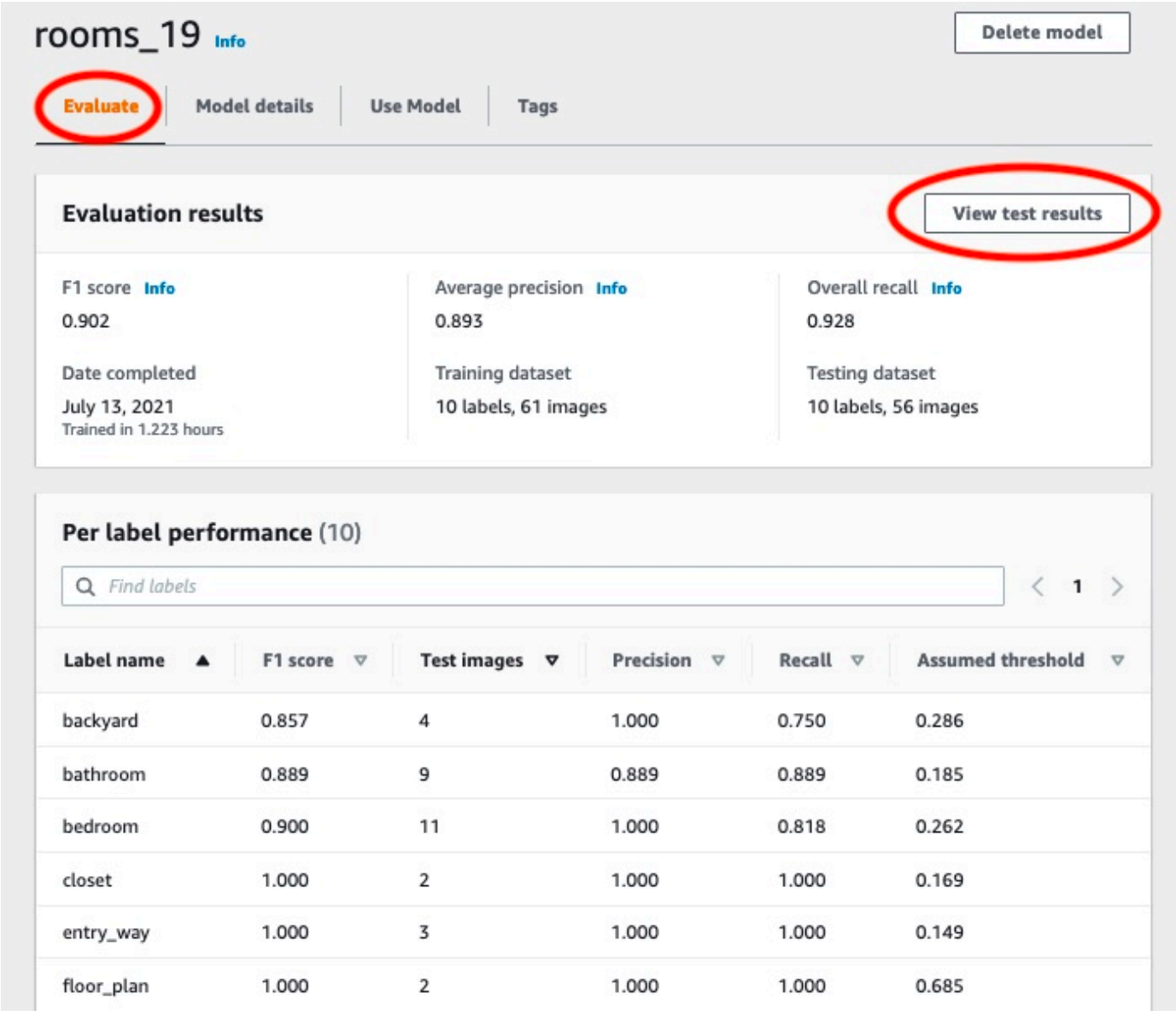
訓練後，訓練資料集是唯讀的。如果您決定改善模型，可以將訓練資料集複製到新的資料集。您可以使用資料集的副本來訓練新版本的模型。

在此步驟中，您需使用主控台來存取主控台中的訓練結果。

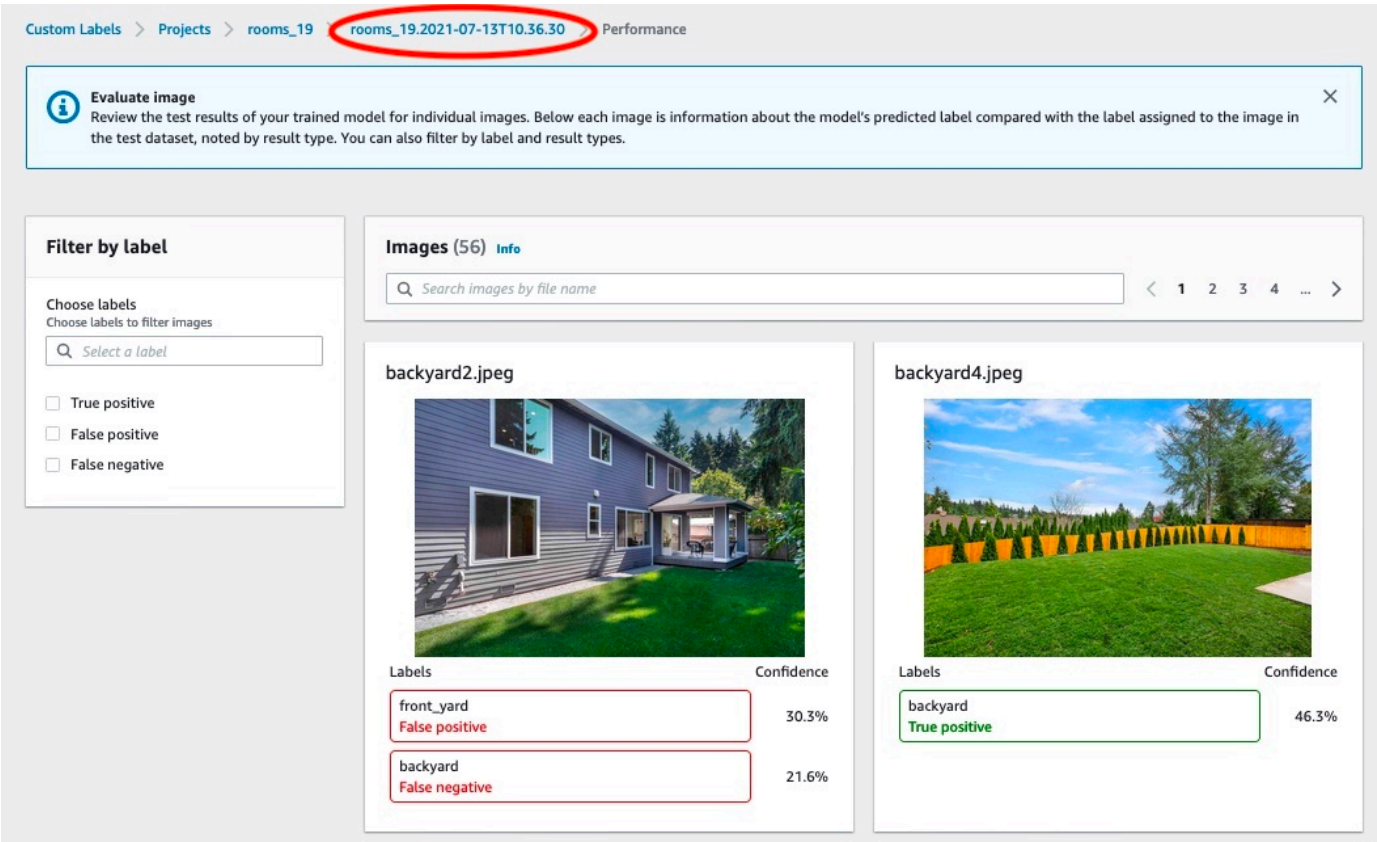
存取評估指標 (主控台)

1. 開啟亞馬遜重新認知主控台，網址為 https://console.aws.amazon.com/rekognition/。

2. 選擇「使用自訂標籤」。
3. 選擇 Get started (開始使用)。
4. 在左側導覽窗格中選擇 Project (專案)。
5. 在「專案」頁面中，選擇包含您要評估的訓練模型的專案。
6. 在模型中選擇您要評估的模型。
7. 選擇「評估」頁標以查看評估結果。如需評估模型的資訊，請參閱改善訓練有素的 Amazon Rekognition 自訂標籤模型。
8. 選擇 [檢視測試結果] 以查看個別測試影像的結果。如需詳細資訊，請參閱評估模型的指標。



9. 檢視測試結果後，選擇要返回模型頁面的專案名稱。



10. 使用指標來評估模型的效能。如需詳細資訊，請參閱改善 Amazon Rekognition 自訂標籤模型。

存取 Amazon Rekognition 自訂標籤評估指標 (SDK)

該DescribeProjectVersions操作提供了對控制台中提供的指標以外的指標的訪問。

與主控台一樣，DescribeProjectVersions可讓您存取下列指標，做為測試結果的摘要資訊，以及作為每個標籤的測試結果：

- 精確度
- 取回
- F1

傳回所有標籤的平均臨界值和個別標籤的臨界值。

DescribeProjectVersions也可讓您存取下列量度，以進行分類和影像偵測 (影像上的物件位置)。

- 用於圖像分類的混淆矩陣。如需詳細資訊，請參閱檢視模型的混淆矩陣。
- 用於圖像檢測的平均平均精度（mAP）。

- 用於圖像檢測的平均平均召回量（Mar）。

DescribeProjectVersions還提供對真正正值、誤報、假負值和真負值的存取。如需詳細資訊，請參閱評估模型的指標。

彙總 F1 分數量度由直接傳回DescribeProjectVersions。您可以從 Amazon S3 儲存貯體中存放的摘要檔案和評估資訊清單檔案存取其他指標。如需詳細資訊，請參閱存取摘要檔案和評估資訊清單快照 (SDK)。

主題

- 摘要檔案
- 評估資訊清單
- 存取摘要檔案和評估資訊清單快照 (SDK)
- 檢視模型的混淆矩陣
- 參考：培訓結果摘要文件

摘要檔案

摘要檔案包含整個模型的評估結果資訊，以及每個標籤的度量。度量標準是精度，召回，F1 得分。也會提供模型的臨界值。摘要檔案位置可從傳回的EvaluationResult物件存取DescribeProjectVersions。如需詳細資訊，請參閱參考：培訓結果摘要文件。

以下是範例摘要檔案。

```
{
  "Version": 1,
  "AggregatedEvaluationResults": {
    "ConfusionMatrix": [
      {
        "GroundTruthLabel": "CAP",
        "PredictedLabel": "CAP",
        "Value": 0.9948717948717949
      },
      {
        "GroundTruthLabel": "CAP",
        "PredictedLabel": "WATCH",
        "Value": 0.008547008547008548
      },
```

```
      {
        "GroundTruthLabel": "WATCH",
        "PredictedLabel": "CAP",
        "Value": 0.1794871794871795
      },
      {
        "GroundTruthLabel": "WATCH",
        "PredictedLabel": "WATCH",
        "Value": 0.7008547008547008
      }
    ],
    "F1Score": 0.9726959470546408,
    "Precision": 0.9719115848331294,
    "Recall": 0.9735042735042735
  },
  "EvaluationDetails": {
    "EvaluationEndTimestamp": "2019-11-21T07:30:23.910943",
    "Labels": [
      "CAP",
      "WATCH"
    ],
    "NumberOfTestingImages": 624,
    "NumberOfTrainingImages": 5216,
    "ProjectVersionArn": "arn:aws:rekognition:us-east-1:nnnnnnnnn:project/my-project/
version/v0/1574317227432"
  },
  "LabelEvaluationResults": [
    {
      "Label": "CAP",
      "Metrics": {
        "F1Score": 0.9794344473007711,
        "Precision": 0.9819587628865979,
        "Recall": 0.9769230769230769,
        "Threshold": 0.9879502058029175
      },
      "NumberOfTestingImages": 390
    },
    {
      "Label": "WATCH",
      "Metrics": {
        "F1Score": 0.9659574468085106,
        "Precision": 0.961864406779661,
        "Recall": 0.9700854700854701,
        "Threshold": 0.014450683258473873
```

```
      },
      "NumberOfTestingImages": 234
    }
  ]
}
```

評估資訊清單

評估資訊清單快照集包含測試結果的詳細資訊。快照包括每個預測的信賴度評分。它還包括與圖像實際分類（真正，真負，假陽性或假陰性）進行比較的預測分類。

這些文件是一個快照，因為只包括可用於測試和培訓的圖像。無法驗證的影像 (例如格式錯誤的影像) 不會包含在資訊清單中。測試快照位置可從傳回的`TestingDataResult`物件存取`DescribeProjectVersions`。訓練快照位置可從傳回的`TrainingDataResult`物件存取`DescribeProjectVersions`。

快照集採用 SageMaker Ground Truth 資訊清單輸出格式，並新增欄位以提供其他資訊，例如偵測的二進位分類結果。下面的代碼片段顯示了其他字段。

```
"rekognition-custom-labels-evaluation-details": {
    "version": 1,
    "is-true-positive": true,
    "is-true-negative": false,
    "is-false-positive": false,
    "is-false-negative": false,
    "is-present-in-ground-truth": true
    "ground-truth-labelling-jobs": ["rekognition-custom-labels-training-job"]
}
```

- version — 資訊清單快照集中`rekognition-custom-labels-evaluation-details`欄位格式的版本。
- is-true-positive... — 根據可信度分數與標籤的最小閾值比較的預測二進位分類。
- is-present-in-ground-truth — 如果模型所做的預測存在於用於培訓的地面真相信息，則為真，否則為假。此值不是基於可信度分數是否超過模型計算的最小閾值。
- ground-truth-labeling-jobs— 清單行中用於訓練的地面真值欄位清單清單清單。

如需有關 SageMaker Ground Truth 資訊清單格式的資訊，請參閱輸出。

以下是測試資訊清單快照的範例，其中顯示影像分類和物件偵測的度量。

```
// For image classification
{
  "source-ref": "s3://test-bucket/dataset/beckham.jpeg",
  "rekognition-custom-labels-training-0": 1,
  "rekognition-custom-labels-training-0-metadata": {
    "confidence": 1.0,
    "job-name": "rekognition-custom-labels-training-job",
    "class-name": "Football",
    "human-annotated": "yes",
    "creation-date": "2019-09-06T00:07:25.488243",
    "type": "groundtruth/image-classification"
  },
  "rekognition-custom-labels-evaluation-0": 1,
  "rekognition-custom-labels-evaluation-0-metadata": {
    "confidence": 0.95,
    "job-name": "rekognition-custom-labels-evaluation-job",
    "class-name": "Football",
    "human-annotated": "no",
    "creation-date": "2019-09-06T00:07:25.488243",
    "type": "groundtruth/image-classification",
    "rekognition-custom-labels-evaluation-details": {
      "version": 1,
      "ground-truth-labelling-jobs": ["rekognition-custom-labels-training-job"],
      "is-true-positive": true,
      "is-true-negative": false,
      "is-false-positive": false,
      "is-false-negative": false,
      "is-present-in-ground-truth": true
    }
  }
}


// For object detection
{
  "source-ref": "s3://test-bucket/dataset/beckham.jpeg",
  "rekognition-custom-labels-training-0": {
    "annotations": [
      {
        "class_id": 0,
        "width": 39,
        "top": 409,
        "height": 63,
```

```
        "left": 712
      },
      ...
    ],
    "image_size": [
      {
        "width": 1024,
        "depth": 3,
        "height": 768
      }
    ]
  },
  "rekognition-custom-labels-training-0-metadata": {
    "job-name": "rekognition-custom-labels-training-job",
    "class-map": {
      "0": "Cap",
      ...
    },
    "human-annotated": "yes",
    "objects": [
      {
        "confidence": 1.0
      },
      ...
    ],
    "creation-date": "2019-10-21T22:02:18.432644",
    "type": "groundtruth/object-detection"
  },
  "rekognition-custom-labels-evaluation": {
    "annotations": [
      {
        "class_id": 0,
        "width": 39,
        "top": 409,
        "height": 63,
        "left": 712
      },
      ...
    ],
    "image_size": [
      {
        "width": 1024,
        "depth": 3,
        "height": 768
```

```
      }
    ]
  },
  "rekognition-custom-labels-evaluation-metadata": {
    "confidence": 0.95,
    "job-name": "rekognition-custom-labels-evaluation-job",
    "class-map": {
      "0": "Cap",
      ...
    },
    "human-annotated": "no",
    "objects": [
      {
        "confidence": 0.95,
        "rekognition-custom-labels-evaluation-details": {
          "version": 1,
          "ground-truth-labelling-jobs": ["rekognition-custom-labels-training-job"],
          "is-true-positive": true,
          "is-true-negative": false,
          "is-false-positive": false,
          "is-false-negative": false,
          "is-present-in-ground-truth": true
        }
      },
      ...
    ],
    "creation-date": "2019-10-21T22:02:18.432644",
    "type": "groundtruth/object-detection"
  }
}
```

存取摘要檔案和評估資訊清單快照 (SDK)

要獲得培訓結果，請致電DescribeProjectVersions。如需範例程式碼，請參閱描述一個模型（SDK）。

測量結果的位置會在回ProjectVersionDescription應中傳回DescribeProjectVersions。

- EvaluationResult— 摘要檔案的位置。
- TestingDataResult— 用於測試的評估資訊清單快照集的位置。

在中傳回 F1 分數和摘要檔案位置EvaluationResult。例如：

```
"EvaluationResult": {
                "F1Score": 1.0,
                "Summary": {
                    "S3Object": {
                        "Bucket": "echo-dot-scans",
                        "Name": "test-output/EvaluationResultSummary-my-echo-dots-
project-v2.json"
                    }
                }
            }
```

評估資訊清單快照集會儲存在您在中指定的 `--output-config`輸入參數中指定的位置訓練模型 (SDK)。

> **Note**
>
> 傳回您收取訓練費用的時間 (秒)`BillableTrainingTimeInSeconds`。

如需 Amazon Rekognition 自訂標籤所傳回之指標的相關資訊，請參閱存取 Amazon Rekognition 自訂標籤評估指標 (SDK)。

檢視模型的混淆矩陣

混淆矩陣可讓您查看模型與模型中其他標籤混淆的標籤。透過使用混淆矩陣，您可以將改進集中在模型上。

在模型評估期間，Amazon Rekognition 自訂標籤會使用測試影像來識別錯誤識別 (混淆) 的標籤，藉此建立混淆矩陣。Amazon Rekognition 自訂標籤只會為分類模型建立混淆矩陣。您可以從 Amazon Rekognition 自訂標籤在模型訓練期間建立的摘要檔案存取分類矩陣。您無法在 Amazon Rekognition 自訂標籤主控台中檢視混淆矩陣。

主題

- 使用混淆矩陣
- 取得模型的混淆矩陣

使用混淆矩陣

下表是「房間」影像分類範例專案的混淆矩陣。列標題是分配給測試圖像的標籤（地面真值標籤）。行標題是模型為測試圖像預測的標籤。每個單元格是對應為地面真值標籤（列）的標籤（行）的預測百分比。例如，對浴室的 67% 的預測被正確標記為浴室。33% 的浴室被錯誤地標記為廚房。當預測的標籤地面真值標籤匹配時，高性能模型具有較高的細胞值。您可以將這些視為從第一個到最後一個預測和地面真值標籤的對角線。如果單元格值為 0，則不會對該單元格的預測標籤進行預測，該標籤應該是單元格的地面真值標籤。

> **Note**
>
> 由於模型不具決定性，因此您訓練 Rooms 專案所得到的混淆矩陣儲存格值可能與下表不同。

混淆矩陣可識別要關注的區域。例如，混淆矩陣顯示模型混淆臥室壁櫥的 50% 的時間。在此情況下，您應該在訓練資料集中新增更多壁櫥和臥室的影像。還要檢查現有的壁櫥和臥室圖像是否正確標記。這應該有助於模型更好地區分兩個標籤。若要將更多影像新增至資料集，請參閱將更多影像新增至資料集。

雖然混淆矩陣很有幫助，但考慮其他指標很重要。例如，100% 的預測正確找到了 floor_plan 標籤，這表明出色的性能。但是，測試資料集只有 2 個帶有 floor_plan 標籤的影像。它還具有 11 個帶有生活空間標籤的圖像。這種不平衡也在訓練數據集中（13 個生活空間圖像和 2 個壁櫥圖像）。為了獲得更準確的評估，請通過添加更多代表性不足的標籤圖像（此示例中的樓層平面圖）來平衡培訓和測試數據集。若要取得每個標籤的測試影像數量，請參閱存取評估指標 (主控台)。

	Ground Truth 標記									
預測標籤	後院	浴室	臥室	壁櫥	入口路	樓板平面圖	前院	廚房	生活空間	露台
後院	75%	0%	0%	0%	0%	0%	33%	0%	0%	0%
浴室	0%	67%	0%	0%	0%	0%	0%	0%	0%	0%
臥室	0%	0%	82%	50%	0%	0%	0%	0%	9%	0%
壁櫥	0%	0%	0%	50%	0%	0%	0%	0%	0%	0%

	Ground Truth 標記									
預測標籤	後院	浴室	臥室	壁櫥	入口路	樓板平面圖	前院	廚房	生活空間	露台
入口路	0%	0%	0%	0%	33%	0%	0%	0%	0%	0%
樓板平面圖	0%	0%	0%	0%	0%	100%	0%	0%	0%	0%
前院	25%	0%	0%	0%	0%	0%	67%	0%	0%	0%
廚房	0%	33%	0%	0%	0%	0%	0%	88%	0%	0%
生活空間	0%	0%	18%	0%	67%	0%	0%	12%	91%	33%
露台	0%	0%	0%	0%	0%	0%	0%	0%	0%	67%

取得模型的混淆矩陣

下列程式碼會使用DescribeProjects和DescribeProjectVersions作業來取得模型的摘要檔案。然後，它會使用摘要檔案來顯示模型的混淆矩陣。

若要顯示模型的混淆矩陣 (SDK)

1. 若您尚未這樣做，請安裝AWS CLI並設定和AWS SDK。如需詳細資訊，請參閱步驟 4：設定 AWS CLI 和 AWS SDKs。
2. 使用下列程式碼來顯示模型的混淆矩陣。提供下列命令列引數：

   - `project_name`— 您要使用的專案名稱。您可以從 Amazon Rekognition 自訂標籤主控台的專案頁面取得專案名稱。
   - `version_name`— 您要使用的模型版本。您可以從 Amazon Rekognition 自訂標籤主控台的專案詳細資訊頁面取得版本名稱。

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

"""
Purpose

Shows how to display the confusion matrix for an Amazon Rekognition Custom labels
 image
classification model.
"""



import json
import argparse
import logging
import boto3
import pandas as pd
from botocore.exceptions import ClientError


logger = logging.getLogger(__name__)


def get_model_summary_location(rek_client, project_name, version_name):
    """
    Get the summary file location for a model.

    :param rek_client: A Boto3 Rekognition client.
    :param project_arn: The Amazon Resource Name (ARN) of the project that contains
 the model.
    :param model_arn: The Amazon Resource Name (ARN) of the model.
    :return: The location of the model summary file.
    """

    try:
        logger.info(
            "Getting summary file for model %s in project %s.", version_name,
 project_name)

        summary_location = ""

        # Get the project ARN from the project name.
```

```
        response = rek_client.describe_projects(ProjectNames=[project_name])

        assert len(response['ProjectDescriptions']) > 0, \
            f"Project {project_name} not found."

        project_arn = response['ProjectDescriptions'][0]['ProjectArn']

        # Get the summary file location for the model.
        describe_response =
 rek_client.describe_project_versions(ProjectArn=project_arn,

 VersionNames=[version_name])
        assert len(describe_response['ProjectVersionDescriptions']) > 0, \
            f"Model {version_name} not found."

        model=describe_response['ProjectVersionDescriptions'][0]

        evaluation_results=model['EvaluationResult']

        summary_location=(f"s3://{evaluation_results['Summary']['S3Object']
['Bucket']}"
                          f"/{evaluation_results['Summary']['S3Object']
['Name']}")

        return summary_location

    except ClientError as err:
        logger.exception(
            "Couldn't get summary file location: %s", err.response['Error']
['Message'])
        raise


def show_confusion_matrix(summary):
    """
    Shows the confusion matrix for an Amazon Rekognition Custom Labels
    image classification model.
    :param summary: The summary file JSON object.
    """
    pd.options.display.float_format = '{:.0%}'.format

    # Load the model summary JSON into a DataFrame.

    summary_df = pd.DataFrame(
```

```
        summary['AggregatedEvaluationResults']['ConfusionMatrix'])

    # Get the confusion matrix.
    confusion_matrix = summary_df.pivot_table(index='PredictedLabel',
                                              columns='GroundTruthLabel',
                                              fill_value=0.0).astype(float)

    # Display the confusion matrix.
    print(confusion_matrix)


def get_summary(s3_resource, summary):
    """
    Gets the summary file.
    : return: The summary file in bytes.
    """
    try:
        summary_bucket, summary_key = summary.replace(
            "s3://", "").split("/", 1)

        bucket = s3_resource.Bucket(summary_bucket)
        obj = bucket.Object(summary_key)
        body = obj.get()['Body'].read()
        logger.info(
            "Got summary file '%s' from bucket '%s'.",
            obj.key, obj.bucket_name)
    except ClientError:
        logger.exception(
            "Couldn't get summary file '%s' from bucket '%s'.",
            obj.key, obj.bucket_name)
        raise
    else:
        return body


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

    parser.add_argument(
        "project_name", help="The ARN of the project in which the model resides."
    )
```

```
    parser.add_argument(
        "version_name", help="The version of the model that you want to describe."
    )


def main():
    """
    Entry point for script.
    """

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

    try:

        # Get the command line arguments.
        parser = argparse.ArgumentParser(usage=argparse.SUPPRESS)
        add_arguments(parser)
        args = parser.parse_args()

        print(
            f"Showing confusion matrix for: {args.version_name} for project
 {args.project_name}.")

        session = boto3.Session(profile_name='custom-labels-access')
        rekognition_client = session.client("rekognition")
        s3_resource = session.resource('s3')

        # Get the summary file for the model.
        summary_location = get_model_summary_location(rekognition_client,
 args.project_name,
                                                      args.version_name
                                                      )
        summary = json.loads(get_summary(s3_resource, summary_location))

        # Check that the confusion matrix is available.
        assert 'ConfusionMatrix' in summary['AggregatedEvaluationResults'], \
            "Confusion matrix not found in summary. Is the model a classification
 model?"

        # Show the confusion matrix.
        show_confusion_matrix(summary)
        print("Done")
```

```
    except ClientError as err:
        logger.exception("Problem showing confusion matrix: %s", err)
        print(f"Problem describing model: {err}")

    except AssertionError as err:
        logger.exception(
            "Error: %s.\n", err)
        print(
            f"Error: {err}\n")


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

參考：培訓結果摘要文件

訓練結果摘要包含可用來評估模型的指標。摘要檔案也可用來在主控台訓練結果頁面中顯示指標。摘要檔案在訓練後存放在 Amazon S3 儲存貯體中。要獲取摘要文件，請致電DescribeProjectVersion。如需範例程式碼，請參閱存取摘要檔案和評估資訊清單快照 (SDK)。

摘要檔案

下列 JSON 是摘要檔案的格式。

EvaluationDetails (第三條)

訓練任務的概觀資訊。這包括模型所屬專案的 ARN (ProjectVersionArn)、訓練完成的日期和時間、評估的模型版本 (EvaluationEndTimestamp)，以及訓練期間偵測到的標籤清單 (Labels)。 還包括用於培訓（ NumberOfTrainingImages ）和評估（ NumberOfTestingImages ）圖像的數量。

AggregatedEvaluationResults (第一部分)

與測試資料集搭配使用時，您可以使用AggregatedEvaluationResults來評估訓練模型的整體效能。包含Precision、Recall和量度的彙總F1Score量度。對於物件偵測 (影像上的物件位置)，會傳回AverageRecall (Mar) 和AveragePrecision (mAP) 度量。針對分類 (影像中的物件類型)，會傳回混淆矩陣量度。

LabelEvaluationResults (第二部分)

您可以使用labelEvaluationResults來評估個別標籤的效能。標籤按每個標籤的 F1 分數排序。包括的量度為PrecisionRecall、F1Score、和Threshold (用於分類)。

檔案名稱的格式如下：EvaluationSummary-ProjectName-VersionName.json

```
{
  "Version": "integer",
  // section-3
  "EvaluationDetails": {
    "ProjectVersionArn": "string",
    "EvaluationEndTimestamp": "string",
    "Labels": "[string]",
    "NumberOfTrainingImages": "int",
    "NumberOfTestingImages": "int"
  },
  // section-1
  "AggregatedEvaluationResults": {
    "Metrics": {
      "Precision": "float",
      "Recall": "float",
      "F1Score": "float",
      // The following 2 fields are only applicable to object detection
      "AveragePrecision": "float",
      "AverageRecall": "float",
      // The following field is only applicable to classification
      "ConfusionMatrix":[
        {
          "GroundTruthLabel": "string",
          "PredictedLabel": "string",
          "Value": "float"
        },
        ...
      ],
    }
  },
  // section-2
  "LabelEvaluationResults": [
    {
      "Label": "string",
      "NumberOfTestingImages", "int",
      "Metrics": {
        "Threshold": "float",
        "Precision": "float",
```

```
        "Recall": "float",
        "F1Score": "float"
      },
    },
    ...
  ]
}
```

改善 Amazon Rekognition 自訂標籤模型

機器學習模型的效能在很大程度上取決於因素，例如自訂標籤的複雜性和變化性 (您感興趣的特定物件和場景)、您提供的訓練資料集的品質和代表性能力，以及用於訓練模型的模型架構和機器學習方法。

Amazon Rekognition 自訂標籤讓這個程序更簡單，而且不需要機器學習專業知識。但是，建置良好模型的程序通常涉及重複資料和模型改進，以達到所需的效能。以下是有關如何改進模型的資訊。

資料

一般而言，您可以利用更高品質的資料來改善模型的品質。使用訓練圖像，清楚地顯示物體或場景，並且不會雜亂不必要的物品。對於物件周圍的邊界方框，請使用訓練影像來展示物件完全可見且不被其他物件遮擋。

請確定您的訓練和測試資料集與您最終將執行推論的映像類型相符。對於只有一些訓練範例的物件 (例如標誌)，您應該在測試影像中的標誌周圍提供邊界方框。這些影像代表或描繪您要在其中本地化物件的案例。

若要將更多影像新增至訓練或測試資料集，請參閱將更多影像新增至資料集。

減少誤報（更好的精確度）

- 首先，檢查是否增加假設的閾值可讓您保持正確的預測，同時減少誤判。在某些時候，由於給定模型的精度和調用之間的權衡，這會減少收益。您無法為標籤設定假設的閾值，但是可以透過將`MinConfidence`輸入參數指定為的高值來達到相同的結果`DetectCustomLabels`。如需詳細資訊，請參閱使用經過培訓的模型分析圖像。
- 您可能會看到一個或多個感興趣的自定義標籤（A）始終與相同類的對象（但不是您感興趣的標籤）（B）混淆。為了幫助您，請將 B 作為對象類標籤添加到訓練數據集中（以及您獲得誤報的圖像）。實際上，您正在通過新的訓練圖像幫助模型學習預測 B 而不是 A。若要將影像新增至訓練資料集，請參閱將更多影像新增至資料集。

- 您可能會發現模型被兩個自定義標籤（A 和 B）混淆-帶有標籤 A 的測試圖像被預測為具有標籤 B，反之亦然。在這種情況下，請先檢查訓練和測試集中是否有標記錯誤的影像。使用資料集庫管理指派給資料集的標籤。如需詳細資訊，請參閱管理標籤。此外，新增更多與此類混淆相關的訓練影像，將有助於重新訓練的模型更好地區分 A 和 B。若要將影像新增至訓練資料集，請參閱將更多影像新增至資料集。

減少假陰性（更好的回憶）

- 對於假設的閾值使用較低的值。您無法為標籤設定假設的閾值，但是可以透過將較低的`MinConfidence`輸入參數指定為來達到相同的結果`DetectCustomLabels`。如需詳細資訊，請參閱使用經過培訓的模型分析圖像。
- 使用更好的範例來建立物件及其顯示影像的各種模型。
- 將您的標籤分成兩個更容易學習的課程。例如，您可能需要好的餅乾，燒焦的 cookie 和破碎的 cookie 來幫助模型更好地學習每個獨特的概念，而不是好的 cookie 和壞餅乾。

執行培訓過的 Amazon Rekognition 自訂標籤模型

當您對模型的效能感到滿意時，您便可以開始使用它。您可使用主控台或 AWS SDK 來啟動和停止模型。主控台還包括您可以使用的範例 SDK 操作。

主題

推論單元

當您啟動模型時，您可以指定模型使用的計算資源 (稱為推論單元) 的數量。

> **Important**
>
> 根據您配置模型執行的方式，您需要根據模型執行的時數以及模型執行時使用的推理單元的數量付費。例如，您使用兩個推論單元啟動模型，並使用該模型 8 小時，則需支付 16 個推論時數的費用（8 小時執行時間 * 2 個推論單元）。如需更多詳細資訊，請參閱 推論時數。如果您沒有明確 停止模型，即使您沒有主動使用模型分析圖像，仍需支付費用。

單一推論單元支援的每秒交易數 (TPS) 會受到以下項目的影響。

- 偵測影像層級標籤（分類）的模型通常比使用邊界框偵測和定位物體（物體偵測）的模型具有更高的 TPS。
- 模型的複雜性。
- 解析度較高的圖像需要更多時間進行分析。
- 圖像中的物體越多，需要更多時間進行分析。
- 較小的圖像比較大的圖像的分析速度更快。
- 以圖像位元組形式傳遞的圖像的分析速度比先將圖像上傳到 Amazon S3 儲存貯體然後引用上傳的圖像要快。作為圖像位元組傳遞的圖像必須小於 4.0 MB。我們建議您在圖像大小小於 4.0 MB 時使用圖像位元組進行近乎即時的圖像處理。例如，從 IP 攝影機擷取的圖像。

- 處理儲存在 Amazon S3 儲存貯體中的圖像比下載圖像、轉換為圖像位元組，然後傳遞圖像位元組進行的分析更快。
- 分析已儲存在 Amazon S3 儲存貯體中的圖像可能比分析作為圖像位元組傳遞的相同圖像更快。如果圖像較大，則尤其如此。

如果呼叫次數 `DetectCustomLabels` 超過模型使用的推論單元總和支援的最大 TPS，Amazon Rekognition 自訂標籤將傳回 `ProvisionedThroughputExceededException` 異常。

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，請使用額外的推論單元。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。有關計算所需推論單元數量的資訊，請參閱計算 Amazon Rekognition 自訂標籤和 Amazon Lookout for Vision 模型的推論單元。如果您想要變更模型的支援輸送量，您有兩種選擇：

手動新增或刪除推論單元

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，並且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，您想要使用每週排程批次呼叫模型。

自動擴展推論單元

如果您的模型必須適應高峰的需求，Amazon Rekognition 自訂標籤可以自動擴展模型使用的推論單元數量。隨著需求的增加，Amazon Rekognition 自訂標籤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 Amazon Rekognition 自訂標籤自動擴展模型的推論單元，請 啟動 模型並使用該 `MaxInferenceUnits` 參數設定可使用的推論單元數目上限。設定推論單元的最大數量可讓您透過限制可用的推論單元數量來管理執行模型的成本。如果您未指定單元數目上限，Amazon Rekognition 自訂標籤不會自動擴展模型，只會使用您開始使用的推論單元數量。如需推論單元數目上限的更多詳細資訊，請參閱 Service Quotas。

您也可以使用 `MinInferenceUnits` 參數指定最小推論單元數量。這可讓您指定模型的最小輸送量，其中單一推論單元代表 1 小時的處理時間。

> **Note**
>
> 您無法使用 Amazon Rekognition 自訂標籤主控台設定推論單元的數量上限。相反，請指定 `StartProjectVersion` 操作的 `MaxInferenceUnits` 輸入參數。

Amazon Rekognition 自訂標籤提供下列 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`	Amazon Rekognition 自訂標籤要縱向擴展或縮減的推論單元數量。
`InServiceInferenceUnits`	模型正在使用的推論單元數目。

如果 `DesiredInferenceUnits` = `InServiceInferenceUnits`，則 Amazon Rekognition 自訂標籤目前不會擴展推論單元的數量。

如果 `DesiredInferenceUnits` > `InServiceInferenceUnits`，則 Amazon Rekognition 自訂標籤將縱向擴展到的 `DesiredInferenceUnits` 值。

如果 `DesiredInferenceUnits` < `InServiceInferenceUnits`，則 Amazon Rekognition 自訂標籤將縮減規模為的 `DesiredInferenceUnits` 值。

如需 Amazon Rekognition 自訂標籤和篩選維度傳回的指標的詳細資訊，請參閱 Rekognition 的CloudWatch 指標。

若要找出您要求的模型推論單元數量上限，請呼叫 `DescribeProjectsVersion` 並檢查回應中的 `MaxInferenceUnits` 欄位。如需範例程式碼，請參閱 描述一個模型（SDK）。

可用區域

Amazon Rekognition 自訂標籤會將推論單元分散到一個 AWS 區域內的多個可用區域，以提供更高的可用性。如需更多詳細資訊，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的情況，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DetectCustomLabels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元所支援的每

秒交易數 (TPS)，則此類呼叫就會成功。AWS 修復可用區域後，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 Rekognition 自訂標籤會在相同的可用區域中自動啟動新的推論單元。模型容量會減少，直到新的推論單元啟動為止。

啟動 Amazon Rekognition 自訂標籤模型

您可以使用主控台或使用作業開始執行 Amazon Rekognition 自訂標籤模型。StartProjectVersion

> **Important**
>
> 您需要根據模型執行的時數以及模型執行時使用的推論單元數量付費。如需詳細資訊，請參閱執行培訓過的 Amazon Rekognition 自訂標籤模型。

啟動模型可能需要幾分鐘才能完成。若要檢查模型就緒的目前狀態，請查看專案或使用的詳細資訊頁面DescribeProjectVersions。

啟動模型後，您可以使用 DetectCustomLabels，使用模型來分析影像。如需詳細資訊，請參閱 使用經過培訓的模型分析圖像。主控台也提供了呼叫 `DetectCustomLabels` 的範例程式碼。

主題

- 啟動 Amazon Rekognition 自訂標籤模型 (主控台)
- 啟動 Amazon Rekognition 自訂標籤模型 (SDK)

啟動 Amazon Rekognition 自訂標籤模型 (主控台)

跟隨以下步驟，開始透過主控台執行 Amazon Rekognition 自訂標籤模型。您可以直接從主控台啟動模型，也可以使用主控台提供的 AWS SDK 程式碼。

啟動模型 (主控台)

1. 開啟 Amazon Rekognition 主控台：https://console.aws.amazon.com/rekognition/。
2. 選擇使用自訂標籤。
3. 選擇 啟動。
4. 在左側導覽視窗中，選擇 專案。

5. 在 專案 資源頁面上，選擇包含要啟動的培訓模型的專案。
6. 在 模型 的區域中，選擇您要啟動的模型。
7. 選擇 使用模型 標籤。
8. 執行以下任意一項：

   Start model using the console

   在 啟動或停止模型 的區域中，執行以下操作：

   1. 選擇您要使用的推論單元數量。如需詳細資訊，請參閱 執行培訓過的 Amazon Rekognition 自訂標籤模型。
   2. 選擇 啟動。
   3. 在 啟動模型 的對話框中，選擇 啟動。

   Start model using the AWS SDK

   在 使用模型 的區域中，執行以下操作：

   1. 選擇 API 程式碼。
   2. 選擇 AWS CLI 或 Python。
   3. 在 啟動模型 中複製範例程式碼。
   4. 使用範例程式碼來啟動模型。如需詳細資訊，請參閱 啟動 Amazon Rekognition 自訂標籤模型 (SDK)。
9. 若要返回專案概述頁面，請選擇頁面頂部的專案名稱。
10. 在 模型 的區域中，檢查模型的狀態。當模型狀態為 執行中 時，您可以使用模型來分析圖像。如需詳細資訊，請參閱 使用經過培訓的模型分析圖像。

啟動 Amazon Rekognition 自訂標籤模型 (SDK)

您可以透過呼叫 StartProjectVersionAPI 並在`ProjectVersionArn`輸入參數中傳遞模型的 Amazon 資源名稱 (ARN) 來啟動模型。您也可指定您要使用的推論單元數量。如需詳細資訊，請參閱 執行培訓過的 Amazon Rekognition 自訂標籤模型。

模型可能需要一段時間才能啟動。本主題中的 Python 和 Java 範例使用等待程式來等待模型啟動。等待程式是一種實用程序方法，用於輪詢特定狀態發生。或者，您可以通過調用來檢查當前狀態DescribeProjectVersions。

啟動模型 (SDK)

1. 如果您尚未執行此作業，請先安裝並配置 AWS CLI 和 AWS SDKs。如需詳細資訊，請參閱 步驟 4：設定 AWS CLI 和 AWS SDKs。
2. 使用以下範例程式碼來啟動模型。

   CLI

   將 `project-version-arn` 的值變更為您要啟動的模型的 ARN。將 `--min-inference-units` 的值變更為您要使用的推論單元數量。（可選）將 `--max-inference-units` 變更為 Amazon Rekognition 自訂標籤可用於自動擴展模型的最大推論單元數量。

   ```
   aws rekognition start-project-version  --project-version-arn model_arn \
      --min-inference-units minimum number of units \
      --max-inference-units maximum number of units \
      --profile custom-labels-access
   ```

   Python

   提供以下命令列參數：

   - `project_arn` — 包含您要啟動的模型專案的 ARN。
   - `model_arn` — 您要啟動的模型 ARN。
   - `min_inference_units` — 您要使用的推論單元的數量。
   - (可選) `--max_inference_units` Amazon Rekognition 自訂標籤可用於自動擴展模型的最大推論單元數量。

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

   """
   Purpose
   Shows how to start running an Amazon Lookout for Vision model.
   """

   import argparse
   import logging
   import boto3
   ```

```
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)


def get_model_status(rek_client, project_arn, model_arn):
    """
    Gets the current status of an Amazon Rekognition Custom Labels model
    :param rek_client: The Amazon Rekognition Custom Labels Boto3 client.
    :param project_name:  The name of the project that you want to use.
    :param model_arn:  The name of the model that you want the status for.
    :return: The model status
    """

    logger.info("Getting status for %s.", model_arn)

    # Extract the model version from the model arn.
    version_name = (model_arn.split("version/", 1)[1]).rpartition('/')[0]

    models = rek_client.describe_project_versions(ProjectArn=project_arn,
                                                  VersionNames=[version_name])

    for model in models['ProjectVersionDescriptions']:

        logger.info("Status: %s", model['StatusMessage'])
        return model["Status"]

    error_message = f"Model {model_arn} not found."
    logger.exception(error_message)
    raise Exception(error_message)


def start_model(rek_client, project_arn, model_arn, min_inference_units,
 max_inference_units=None):
    """
    Starts the hosting of an Amazon Rekognition Custom Labels model.
    :param rek_client: The Amazon Rekognition Custom Labels Boto3 client.
    :param project_name:  The name of the project that contains 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: The number of inference units to use for auto-
scaling
    the model. If not supplied, auto-scaling does not happen.
```

```
    """

    try:
        # Start the model
        logger.info(f"Starting model: {model_arn}. Please wait....")

        if max_inference_units is None:
            rek_client.start_project_version(ProjectVersionArn=model_arn,

 MinInferenceUnits=int(min_inference_units))
        else:
            rek_client.start_project_version(ProjectVersionArn=model_arn,
                                             MinInferenceUnits=int(
                                                 min_inference_units),

 MaxInferenceUnits=int(max_inference_units))

        # Wait for the model to be in the running state
        version_name = (model_arn.split("version/", 1)[1]).rpartition('/')[0]
        project_version_running_waiter = rek_client.get_waiter(
            'project_version_running')
        project_version_running_waiter.wait(
            ProjectArn=project_arn, VersionNames=[version_name])

        # Get the running status
        return get_model_status(rek_client, project_arn, model_arn)

    except ClientError as err:
        logger.exception("Client error: Problem starting model: %s", err)
        raise


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

    parser.add_argument(
        "project_arn", help="The ARN of the project that contains that the model
 you want to start."
    )
    parser.add_argument(
        "model_arn", help="The ARN of the model that you want to start."
```

```
    )
    parser.add_argument(
        "min_inference_units", help="The minimum number of inference units to
 use."
    )
    parser.add_argument(
        "--max_inference_units",  help="The maximum number of inference units to
 use for auto-scaling the model.", 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()

        # Start the model.
        session = boto3.Session(profile_name='custom-labels-access')
        rekognition_client = session.client("rekognition")

        status = start_model(rekognition_client,
                             args.project_arn, args.model_arn,
                             args.min_inference_units,
                             args.max_inference_units)

        print(f"Finished starting model: {args.model_arn}")
        print(f"Status: {status}")

    except ClientError as err:
        error_message = f"Client error: Problem starting model: {err}"
        logger.exception(error_message)
        print(error_message)

    except Exception as err:
        error_message = f"Problem starting model:{err}"
        logger.exception(error_message)
        print(error_message)
```

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

Java V2

提供以下命令列參數：

- project_arn — 包含您要啟動的模型專案的 ARN。
- model_arn — 您要啟動的模型 ARN。
- min_inference_units — 您要使用的推論單元的數量。
- (可選) max_inference_units — Amazon Rekognition 自訂標籤可用於自動擴展模型的最大推論單元數量。如果您未有指定值，則不會進行自動擴展。

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

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import
 software.amazon.awssdk.services.rekognition.model.DescribeProjectVersionsRequest;
import
 software.amazon.awssdk.services.rekognition.model.DescribeProjectVersionsResponse;
import
 software.amazon.awssdk.services.rekognition.model.ProjectVersionDescription;
import software.amazon.awssdk.services.rekognition.model.ProjectVersionStatus;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import
 software.amazon.awssdk.services.rekognition.model.StartProjectVersionRequest;
import
 software.amazon.awssdk.services.rekognition.model.StartProjectVersionResponse;
import software.amazon.awssdk.services.rekognition.waiters.RekognitionWaiter;

import java.util.Optional;
import java.util.logging.Level;
```

```
import java.util.logging.Logger;

public class StartModel {

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



    public static int findForwardSlash(String modelArn, int n) {

        int start = modelArn.indexOf('/');
        while (start >= 0 && n > 1) {
            start = modelArn.indexOf('/', start + 1);
            n -= 1;
        }
        return start;

    }

    public static void startMyModel(RekognitionClient rekClient, String
 projectArn, String modelArn,
            Integer minInferenceUnits, Integer maxInferenceUnits
            ) throws Exception, RekognitionException {

        try {

            logger.log(Level.INFO, "Starting model: {0}", modelArn);

            StartProjectVersionRequest startProjectVersionRequest = null;

            if (maxInferenceUnits == null) {
                startProjectVersionRequest =
 StartProjectVersionRequest.builder()
                    .projectVersionArn(modelArn)
                    .minInferenceUnits(minInferenceUnits)
                    .build();
            }
            else {
                startProjectVersionRequest =
 StartProjectVersionRequest.builder()
                        .projectVersionArn(modelArn)
                        .minInferenceUnits(minInferenceUnits)
                        .maxInferenceUnits(maxInferenceUnits)
```

```
                        .build();

            }

            StartProjectVersionResponse response =
rekClient.startProjectVersion(startProjectVersionRequest);

            logger.log(Level.INFO, "Status: {0}", response.statusAsString() );


            // Get the model version

            int start = findForwardSlash(modelArn, 3) + 1;
            int end = findForwardSlash(modelArn, 4);

            String versionName = modelArn.substring(start, end);


            // wait until model starts

            DescribeProjectVersionsRequest describeProjectVersionsRequest =
DescribeProjectVersionsRequest.builder()
                    .versionNames(versionName)
                    .projectArn(projectArn)
                    .build();

            RekognitionWaiter waiter = rekClient.waiter();

            WaiterResponse<DescribeProjectVersionsResponse> waiterResponse =
waiter

.waitUntilProjectVersionRunning(describeProjectVersionsRequest);

            Optional<DescribeProjectVersionsResponse> optionalResponse =
waiterResponse.matched().response();

            DescribeProjectVersionsResponse describeProjectVersionsResponse =
optionalResponse.get();

            for (ProjectVersionDescription projectVersionDescription :
describeProjectVersionsResponse
                    .projectVersionDescriptions()) {
                if(projectVersionDescription.status() ==
ProjectVersionStatus.RUNNING) {
```

```
                    logger.log(Level.INFO, "Model is running" );

                }
                else {
                    String error = "Model training failed: " +
projectVersionDescription.statusAsString() + " "
                            + projectVersionDescription.statusMessage() + " " +
modelArn;
                    logger.log(Level.SEVERE, error);
                    throw new Exception(error);
                }

            }


        } catch (RekognitionException e) {
            logger.log(Level.SEVERE, "Could not start model: {0}",
e.getMessage());
            throw e;
        }

    }

    public static void main(String[] args) {

        String modelArn = null;
        String projectArn = null;
        Integer minInferenceUnits = null;
        Integer maxInferenceUnits = null;




        final String USAGE = "\n" + "Usage: " + "<project_name> <version_name>
<min_inference_units> <max_inference_units>\n\n" + "Where:\n"
                + "   project_arn - The ARN of the project that contains the
model that you want to start. \n\n"
                + "   model_arn - The ARN of the model version that you want to
start.\n\n"
                + "   min_inference_units - The number of inference units to
start the model with.\n\n"
                + "   max_inference_units - The maximum number of inference
units that Custom Labels can use to "
```

```
                + "   automatically scale the model. If the value is null,
 automatic scaling doesn't happen.\n\n";

        if (args.length < 3  || args.length >4) {
            System.out.println(USAGE);
            System.exit(1);
        }

        projectArn = args[0];
        modelArn = args[1];
        minInferenceUnits=Integer.parseInt(args[2]);

        if (args.length == 4) {
            maxInferenceUnits = Integer.parseInt(args[3]);
        }

        try {


            // Get the Rekognition client.
            RekognitionClient rekClient = RekognitionClient.builder()
            .credentialsProvider(ProfileCredentialsProvider.create("custom-
labels-access"))
            .region(Region.US_WEST_2)
            .build();

            // Start the model.
            startMyModel(rekClient, projectArn, modelArn, minInferenceUnits,
 maxInferenceUnits);


            System.out.println(String.format("Model started: %s", modelArn));

            rekClient.close();

        } catch (RekognitionException rekError) {
            logger.log(Level.SEVERE, "Rekognition client error: {0}",
 rekError.getMessage());
            System.exit(1);
        } catch (Exception rekError) {
            logger.log(Level.SEVERE, "Error: {0}", rekError.getMessage());
            System.exit(1);
        }
```

```
    }
}
```

停止 Amazon Rekognition 自訂標籤模型

您可以使用主控台或使用作業停止執行 Amazon Rekognition 自訂標籤模型。StopProjectVersion

主題

- 停止 Amazon Rekognition 自訂標籤模型 (主控台)
- 停止 Amazon Rekognition 自訂標籤模型 (SDK)

停止 Amazon Rekognition 自訂標籤模型 (主控台)

請使用以下步驟來停止執行中的 Amazon Rekognition 自訂標籤模型。您可以直接從主控台停止模型，或使用主控台提供的 AWS SDK 程式碼。

停止模型 (主控台)

1. 開啟 Amazon Rekognition 主控台：https://console.aws.amazon.com/rekognition/。
2. 選擇使用自訂標籤。
3. 選擇 啟動。
4. 在左側導覽視窗中，選擇 專案。
5. 在 專案 頁面中，選擇包含要停止的培訓模型的專案。
6. 在 模型 的區域中，選擇您要停止的模型。
7. 選擇 使用模型 標籤。
8. Stop model using the console
    1. 在 啟動或停止模型 的區域中，選擇 停止。
    2. 在 停止模型 的對話框中，輸入 停止 以確認您要停止模型。
    3. 選擇 停止 以停止模型。

Stop model using the AWS SDK

在 使用模型 的區域中，執行下列操作：

1. 選擇 API 程式碼。
2. 選擇 AWS CLI 或 Python。
3. 在 停止模型 中複製範例程式碼。
4. 使用範例程式碼來停止模型。如需詳細資訊，請參閱 停止 Amazon Rekognition 自訂標籤模型 (SDK)。

9. 在頁面頂端選擇您的專案名稱，以返回專案概述頁面。
10. 在 模型 的區域中，檢查模型的狀態。當模型狀態顯示為 已停止 時，則表示模型已經停止。

停止 Amazon Rekognition 自訂標籤模型 (SDK)

您可以通過調用 StopProjectVersionAPI 並在`ProjectVersionArn`輸入參數中傳遞模型的 Amazon 資源名稱（ARN）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 `DescribeProjectVersions`。

停止模型 (SDK)

1. 如果您尚未執行此作業，請先安裝並配置 AWS CLI 和 AWS SDKs。如需更多詳細資訊，請參閱 步驟 4：設定 AWS CLI 和 AWS SDKs。
2. 使用下列範例程式碼來停止執行中的模型。

CLI

將 `project-version-arn` 的值變更為您要停止的模型版本的 ARN。

```
aws rekognition stop-project-version --project-version-arn "model arn" \
  --profile custom-labels-access
```

Python

以下範例會停止已在執行中的模型。

提供以下命令列參數：

- project_arn — 包含您要停止的模型的專案的 ARN。
- model_arn — 您要停止的模型 ARN。

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

"""
Purpose
Shows how to stop a running Amazon Lookout for Vision model.
"""

import argparse
import logging
import time
import boto3

from botocore.exceptions import ClientError


logger = logging.getLogger(__name__)


def get_model_status(rek_client, project_arn, model_arn):
    """
    Gets the current status of an Amazon Rekognition Custom Labels model
    :param rek_client: The Amazon Rekognition Custom Labels Boto3 client.
    :param project_name:  The name of the project that you want to use.
    :param model_arn:  The name of the model that you want the status for.
    """

    logger.info ("Getting status for %s.", model_arn)

    # Extract the model version from the model arn.
    version_name=(model_arn.split("version/",1)[1]).rpartition('/')[0]

    # Get the model status.
    models=rek_client.describe_project_versions(ProjectArn=project_arn,
    VersionNames=[version_name])

    for model in models['ProjectVersionDescriptions']:
        logger.info("Status: %s",model['StatusMessage'])
```

```
        return model["Status"]

    # No model found.
    logger.exception("Model %s not found.", model_arn)
    raise Exception("Model %s not found.", model_arn)


def stop_model(rek_client, project_arn, model_arn):
    """
    Stops a running Amazon Rekognition Custom Labels Model.
    :param rek_client: The Amazon Rekognition Custom Labels Boto3 client.
    :param project_arn: The ARN of the project that you want to stop running.
    :param model_arn:  The ARN of the model (ProjectVersion) that you want to
 stop running.
    """

    logger.info("Stopping model: %s", model_arn)

    try:
        # Stop the model.
        response=rek_client.stop_project_version(ProjectVersionArn=model_arn)

        logger.info("Status: %s", response['Status'])

        # stops when hosting has stopped or failure.
        status = ""
        finished = False

        while finished is False:

            status=get_model_status(rek_client, project_arn, model_arn)

            if status == "STOPPING":
                logger.info("Model stopping in progress...")
                time.sleep(10)
                continue
            if status == "STOPPED":
                logger.info("Model is not running.")
                finished = True
                continue

            error_message = f"Error stopping model. Unexepected state: {status}"
            logger.exception(error_message)
            raise Exception(error_message)
```

```
        logger.info("finished. Status %s", status)
        return status

    except ClientError as err:
        logger.exception("Couldn't stop model - %s: %s",
           model_arn,err.response['Error']['Message'])
        raise


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

    parser.add_argument(
        "project_arn", help="The ARN of the project that contains the model that
 you want to stop."
    )
    parser.add_argument(
        "model_arn", help="The ARN of the model that you want to stop."
    )

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()

        # Stop the model.
        session = boto3.Session(profile_name='custom-labels-access')
        rekognition_client = session.client("rekognition")

        status=stop_model(rekognition_client, args.project_arn, args.model_arn)

        print(f"Finished stopping model: {args.model_arn}")
        print(f"Status: {status}")
```

```
    except ClientError as err:
        logger.exception("Problem stopping model:%s",err)
        print(f"Failed to stop model: {err}")

    except Exception as err:
        logger.exception("Problem stopping model:%s", err)
        print(f"Failed to stop model: {err}")

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

Java V2

提供以下命令列參數：

- project_arn — 包含您要停止的模型的專案的 ARN。
- model_arn — 您要停止的模型 ARN。

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

package com.example.rekognition;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import
 software.amazon.awssdk.services.rekognition.model.DescribeProjectVersionsRequest;
import
 software.amazon.awssdk.services.rekognition.model.DescribeProjectVersionsResponse;
import
 software.amazon.awssdk.services.rekognition.model.ProjectVersionDescription;
import software.amazon.awssdk.services.rekognition.model.ProjectVersionStatus;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import
 software.amazon.awssdk.services.rekognition.model.StopProjectVersionRequest;
import
 software.amazon.awssdk.services.rekognition.model.StopProjectVersionResponse;
```

```
import java.util.logging.Level;
import java.util.logging.Logger;

public class StopModel {

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

    public static int findForwardSlash(String modelArn, int n) {

        int start = modelArn.indexOf('/');
        while (start >= 0 && n > 1) {
            start = modelArn.indexOf('/', start + 1);
            n -= 1;
        }
        return start;

    }

    public static void stopMyModel(RekognitionClient rekClient, String
 projectArn, String modelArn)
            throws Exception, RekognitionException {

        try {

            logger.log(Level.INFO, "Stopping {0}", modelArn);

            StopProjectVersionRequest stopProjectVersionRequest =
 StopProjectVersionRequest.builder()
                    .projectVersionArn(modelArn).build();

            StopProjectVersionResponse response =
 rekClient.stopProjectVersion(stopProjectVersionRequest);

            logger.log(Level.INFO, "Status: {0}", response.statusAsString());

            // Get the model version

            int start = findForwardSlash(modelArn, 3) + 1;
            int end = findForwardSlash(modelArn, 4);

            String versionName = modelArn.substring(start, end);
```

```
            // wait until model stops

            DescribeProjectVersionsRequest describeProjectVersionsRequest =
DescribeProjectVersionsRequest.builder()
                    .projectArn(projectArn).versionNames(versionName).build();

            boolean stopped = false;

            // Wait until create finishes

            do {

                DescribeProjectVersionsResponse describeProjectVersionsResponse
= rekClient

.describeProjectVersions(describeProjectVersionsRequest);

                for (ProjectVersionDescription projectVersionDescription :
describeProjectVersionsResponse
                        .projectVersionDescriptions()) {

                    ProjectVersionStatus status =
projectVersionDescription.status();

                    logger.log(Level.INFO, "stopping model: {0} ", modelArn);

                    switch (status) {

                    case STOPPED:
                        logger.log(Level.INFO, "Model stopped");
                        stopped = true;
                        break;

                    case STOPPING:
                        Thread.sleep(5000);
                        break;

                    case FAILED:
                        String error = "Model stopping failed: " +
projectVersionDescription.statusAsString() + " "
                                + projectVersionDescription.statusMessage() + "
" + modelArn;
                        logger.log(Level.SEVERE, error);
                        throw new Exception(error);
```

```
                    default:
                        String unexpectedError = "Unexpected stopping state: "
                                + projectVersionDescription.statusAsString() + "
 "
                                + projectVersionDescription.statusMessage() + "
 " + modelArn;
                        logger.log(Level.SEVERE, unexpectedError);
                        throw new Exception(unexpectedError);
                    }
                }

            } while (stopped == false);

        } catch (RekognitionException e) {
            logger.log(Level.SEVERE, "Could not stop model: {0}",
 e.getMessage());
            throw e;
        }

    }

    public static void main(String[] args) {

        String modelArn = null;
        String projectArn = null;


        final String USAGE = "\n" + "Usage: " + "<project_name> <version_name>\n
\n" + "Where:\n"
                + "   project_arn - The ARN of the project that contains the
 model that you want to stop. \n\n"
                + "   model_arn - The ARN of the model version that you want to
 stop.\n\n";

        if (args.length != 2) {
            System.out.println(USAGE);
            System.exit(1);
        }

        projectArn = args[0];
        modelArn = args[1];
```

```
        try {

            // Get the Rekognition client.
            RekognitionClient rekClient = RekognitionClient.builder()
            .credentialsProvider(ProfileCredentialsProvider.create("custom-
labels-access"))
            .region(Region.US_WEST_2)
            .build();

            // Stop model
            stopMyModel(rekClient, projectArn, modelArn);

            System.out.println(String.format("Model stopped: %s", modelArn));

            rekClient.close();

        } catch (RekognitionException rekError) {
            logger.log(Level.SEVERE, "Rekognition client error: {0}",
 rekError.getMessage());
            System.exit(1);
        } catch (Exception rekError) {
            logger.log(Level.SEVERE, "Error: {0}", rekError.getMessage());
            System.exit(1);
        }

    }

}
```

報告使用的執行持續時間和推論單元

如果您在 2022 年 8 月之後訓練並啟動模型，則可以使用 `InServiceInferenceUnits` Amazon CloudWatch 指標來確定模型執行了多少小時，以及在這些時間內使用的推論單元數量。

> **Note**
>
> 如果一個AWS區域中只有一個模型，您也可以追蹤對`StartprojectVersion`和傳入的成功呼叫，以取得模型的執行時`StopProjectVersion`間 CloudWatch。如果您在 AWS 區域中執行多個模型，則此方法不起作用，因為指標不包含有關模型的資訊。

或者，您可以使用AWS CloudTrail來追蹤`StartProjectVersion`和的呼叫`StopProjectVersion` (其中包括事件歷史記錄`requestParameters`欄位中的模型 ARN)。CloudTrail 活動限制為 90 天，但您可以在CloudTrail湖泊中存放長達 7 年的活動。

以下步驟會針對下列項目建立圖表：

- 模型已執行的時數。
- 模型使用的推論單元數量。

您最多可以選擇過去 15 個月的時間段。如需有關指標保留的更多詳細資訊，請參閱 指標保留。

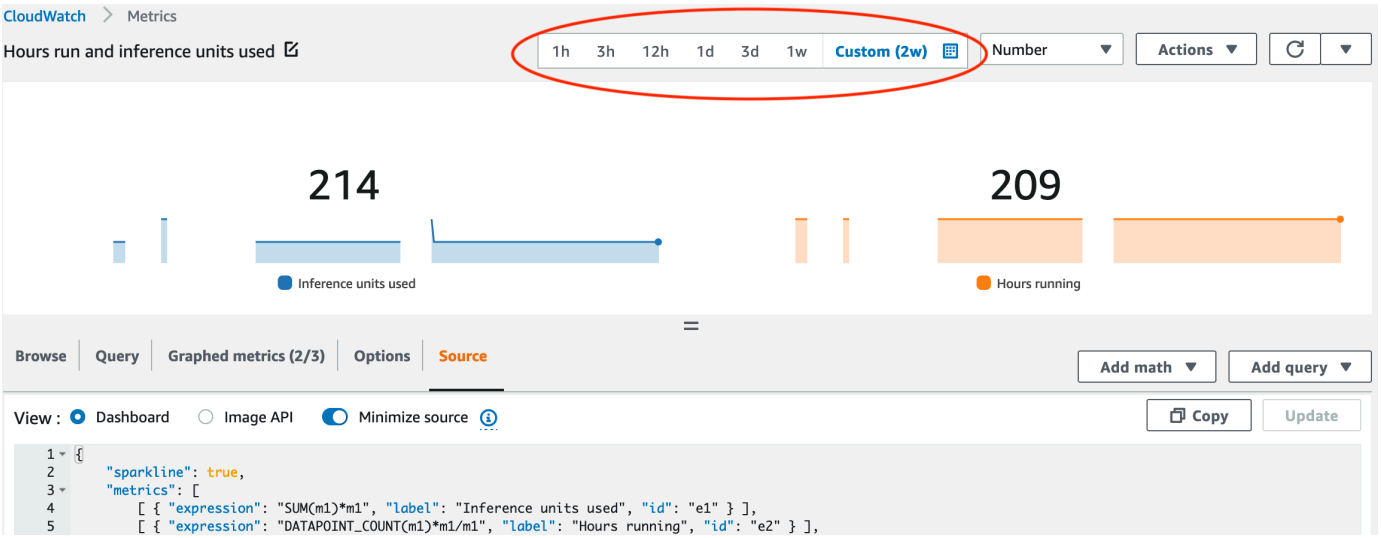
決定模型使用的模型持續時間和推論單元

1. 請登入AWS Management Console並開啟 CloudWatch 主控台，網址為 https://console.aws.amazon.com/cloudwatch/。
2. 在左側導覽視窗中，選擇 指標 下的 所有指標。
3. 在下方視窗中，選擇 來源 標籤。
4. 確定已選取 儀表板 按鈕。
5. 在編輯框中，將現有 JSON 替換為以下 JSON。更改以下的值：
   - `Project_Name` — 包含要繪製圖表的模型專案。
   - `Version_Name` — 您想要繪製圖表的模型版本。
   - `AWS_Region` — 包含模型的 AWS 區域。檢查頁面頂端導覽列中的 CloudWatch [AWS地區] 選取器，確認主控台位於相同的 [區域]。根據需要進行更新。

```
{
    "sparkline": true,
    "metrics": [
        [
            {
                "expression": "SUM(m1)*m1",
                "label": "Inference units used",
                "id": "e1"
            }
        ],
        [
```

```
            {
                "expression": "DATAPOINT_COUNT(m1)*m1/m1",
                "label": "Hours running",
                "id": "e2"
            }
        ],
        [
            "AWS/Rekognition",
            "InServiceInferenceUnits",
            "ProjectName",
            "Project_Name",
            "VersionName",
            "Version_Name",
            {
                "id": "m1",
                "visible": false
            }
        ]
    ],
    "view": "singleValue",
    "stacked": false,
    "region": "AWS_Region",
    "stat": "Average",
    "period": 3600,
    "title": "Hours run and inference units used"
}
```

6. 選擇 更新。
7. 在頁面頂部，選擇時間軸。您應該會看到時間軸內使用的推論單元數量和執行的時數。圖表中的間隙表示模型未執行的時間。



8. (可選) 透過選擇 動作，然後 新增至儀表板 - 改進，將圖表新增至儀表板。

使用經過培訓的模型分析圖像

若要使用訓練有素的 Amazon Rekognition 自訂標籤模型分析影像，您可以呼叫 API。DetectCustomLabelsDetectCustomLabels 的結果是圖像包含了特定物體、場景或概念的預測。

若要呼叫 DetectCustomLabels，您需指定以下內容：

- 您想要使用的 Amazon Rekognition 自訂標籤模型的 Amazon Resource Name (ARN)。
- 您希望使用模型用來進行預測的圖像。您可以將輸入圖像提供為圖像位元組陣列 (base64 編碼影像位元組) 或 Amazon S3 物體。如需更多詳細資訊，請參閱 圖像。

自訂標籤會在 自訂標籤 物體的陣列中傳回。每個自訂標籤代表圖像中的單一物體、場景或概念。自訂標籤包括：

- 圖像中找到的物體、場景或概念的標籤。
- 在圖像中找到的物體的邊界框。邊界框座標會顯示物體在來源圖像上的位置。座標值是整個圖像大小的比率。如需詳細資訊，請參閱BoundingBox。 DetectCustomLabels只有在模型經過訓練以偵測物件位置時，才會傳回邊界方框。
- Amazon Rekognition 自訂標籤對標籤和邊界框的準確性有信心。

若要根據偵測信賴度篩選標籤，請指定符合 MinConfidence 所需信賴等級的值。例如，您需要對預測非常有信心，請為 MinConfidence 指定一個較高的值。若要取得所有標籤，而不需管信賴度，請將 MinConfidence 的值設為 0。

模型的性能部分是透過模型培訓期間計算的召回率和精確度指標來衡量的。如需更多詳細資訊，請參閱評估模型的指標。

若要提高模型的精確度，請將 MinConfidence 設定為較高的值。如需更多詳細資訊，請參閱 減少誤報（更好的精確度）。

若要提高模型的召回率，請將 MinConfidence 設定為較低的值。如需更多詳細資訊，請參閱 減少假陰性（更好的回憶）。

如果您沒有為 MinConfidence 設定指定的值，Amazon Rekognition 自訂標籤將根據該標籤的假定臨界值傳回標籤。如需更多詳細資訊，請參閱 假設閾值。您可以從模型的培訓結果中取得標籤的假定臨界值。如需更多詳細資訊，請參閱 訓練模型（控制台）。

透過使用 MinConfidence 輸入參數，即可指定呼叫所需的臨界值。回應中不會傳回偵測到信賴度低於 MinConfidence 的標籤。此外，標籤的假定臨界值不會影響回應中包含的標籤。

> **Note**
>
> Amazon Rekognition 自訂標籤指標以 0-1 之間的浮點值表示假定臨界值。MinConfidence 的範圍會將臨界值標準化為百分比值 (0-100)。來自的信賴度回應 DetectCustomLabels 也會以百分比傳回。

您可能想要為特定標籤指定臨界值。例如，當標籤 A 可接受精確度指標的結果，但標籤 B 卻無法接受。當指定不同的臨界值 (MinConfidence) 時，請考慮下列事項：

- 如果您只對單一標籤 (A) 感興趣，請將 MinConfidence 的值設定為所需的臨界值。在回應中，只有當信賴度大於 MinConfidence 時，才會傳回標籤 A 的預測 (以及其他標籤)。您需要過濾掉傳回的任何其他標籤。
- 如果您要將不同的臨界值套用至多個標籤，請執行以下操作：
  1. 將 MinConfidence 的值設為 0。無論偵測的信賴度如何，0 值可以確保能傳回所有標籤。
  2. 針對傳回的每個標籤，透過檢查標籤信賴度是否大於您想要的標籤臨界值，以套用所需的臨界值。

如需更多詳細資訊，請參閱 改善訓練有素的 Amazon Rekognition 自訂標籤模型。

如果您發現 DetectCustomLabels 傳回的信賴度值太低，請考慮重新培訓模型。如需更多詳細資訊，請參閱 訓練 Amazon Rekognition 自訂標籤模型。您可以透過指定 MaxResults 輸入參數來限制 DetectCustomLabels 傳回的自訂標籤的數量。傳回的信賴度結果將會由高至低開始排序。

如需其他呼叫 DetectCustomLabels 的範例，請參閱 範例。

如需有關保護 DetectCustomLabels 的資訊，請參閱 安全DetectCustomLabels。

偵測自訂標籤 (API)

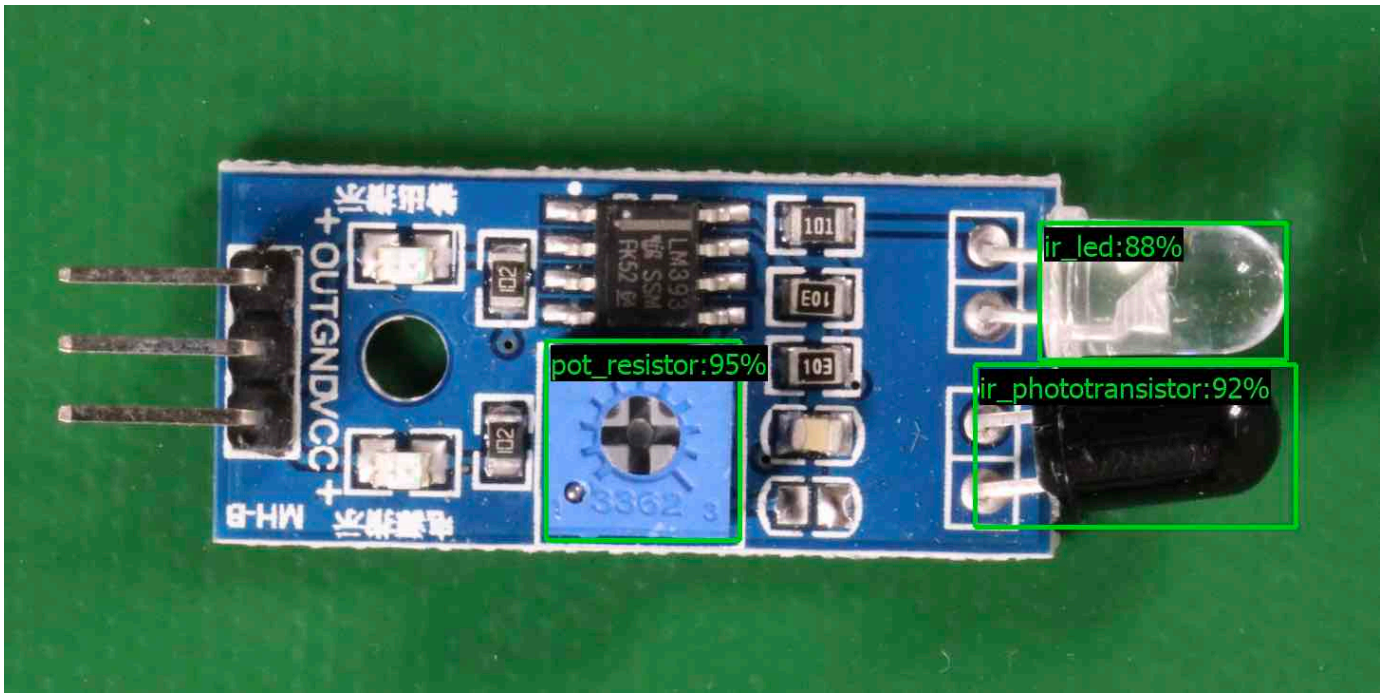
1. 如果您尚未執行：

   a. 請確保您擁有 DetectCustomLabels 和 AmazonS3ReadOnlyAccess 的權限。如需詳細資訊，請參閱 設定 SDK 權限。

   b. 安裝和設定 AWS CLI 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 SDKs。

2. 培訓並部署好您的模型。如需更多詳細資訊，請參閱 建立 Amazon Rekognition 型。
3. 確保呼叫 DetectCustomLabels 的使用者可以存取您在步驟 2 中使用的模型。如需更多詳細資訊，請參閱 安全DetectCustomLabels。
4. 將您要分析的圖像上傳到 S3 儲存貯體。

   如需說明，請參閱 Amazon 簡單儲存服務使用者指南 中的 將物體上傳至 Amazon S3。Python、Java 和 Java 2 範例也向您展示了如何使用本機圖像文檔透過原始位元組傳遞圖像。檔案必須小於 4 MB。
5. 使用以下範例來呼叫 DetectCustomLabels 操作。Python 和 Java 範例中會顯示圖像並疊加分析結果，類似於下圖所示。



AWS CLI

此 AWS CLI 命令會顯示 DetectCustomLabels CLI 作業的 JSON 輸出。變更以下輸入參數的值。

- bucket 為您在步驟 4 中使用的 Amazon S3 儲存貯體的名稱。
- image 為您在步驟 4 中上傳的載入圖像的檔案名稱。
- projectVersionArn 為您要使用的模型的 ARN。

```
aws rekognition detect-custom-labels --project-version-arn model_arn \
```

```
    --image '{"S3Object":{"Bucket":"bucket","Name":"image"}}' \
    --min-confidence 70 \
    --profile custom-labels-access
```

Python

以下的範例程式碼顯示圖像中找到的邊界框和圖像層級標籤。

若要分析本機圖像，請執行程式並提供以下命令列參數：

- 您要用來分析圖像的模型的 ARN。
- 本機圖像檔案的名稱和位置。

若要分析儲存在 Amazon S3 儲存貯體中的圖像，請執行程式並提供以下命令列參數：

- 您要用來分析圖像的模型的 ARN。
- 您在步驟 4 中使用的 Amazon S3 儲存貯體中圖像的名稱和位置。
- `--bucket ######` — 您在步驟 4 中使用的 Amazon S3 儲存貯體。

請注意，此範例假設您的枕頭版本大於 = 8.0.0。

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Amazon Rekognition Custom Labels detection example used in the service
 documentation:
https://docs.aws.amazon.com/rekognition/latest/customlabels-dg/detecting-custom-
labels.html
Shows how to detect custom labels by using an Amazon Rekognition Custom Labels
 model.
The image can be stored on your local computer or in an Amazon S3 bucket.
"""

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

```
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)


def analyze_local_image(rek_client, model, photo, min_confidence):
    """
    Analyzes an image stored as a local file.
    :param rek_client: The Amazon Rekognition Boto3 client.
    :param s3_connection: The Amazon S3 Boto3 S3 connection object.
    :param model: The ARN of the Amazon Rekognition Custom Labels model that you
 want to use.
    :param photo: The name and file path of the photo that you want to analyze.
    :param min_confidence: The desired threshold/confidence for the call.
    """

    try:
        logger.info("Analyzing local file: %s", photo)
        image = Image.open(photo)
        image_type = Image.MIME[image.format]

        if (image_type == "image/jpeg" or image_type == "image/png") is False:
            logger.error("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()

        response = rek_client.detect_custom_labels(Image={'Bytes': image_bytes},
                                                   MinConfidence=min_confidence,
                                                   ProjectVersionArn=model)

        show_image(image, response)
        return len(response['CustomLabels'])

    except ClientError as client_err:
        logger.error(format(client_err))
        raise
    except FileNotFoundError as file_error:
```

```
        logger.error(format(file_error))
        raise


def analyze_s3_image(rek_client, s3_connection, model, bucket, photo,
 min_confidence):
    """
    Analyzes an image stored in the specified S3 bucket.
    :param rek_client: The Amazon Rekognition Boto3 client.
    :param s3_connection: The Amazon S3 Boto3 S3 connection object.
    :param model: The ARN of the Amazon Rekognition Custom Labels model that you
 want to use.
    :param bucket: The name of the S3 bucket that contains the image that you
 want to analyze.
    :param photo: The name of the photo that you want to analyze.
    :param min_confidence: The desired threshold/confidence for the call.
    """

    try:
        # Get image from S3 bucket.

        logger.info("analyzing bucket: %s image: %s", bucket, photo)
        s3_object = s3_connection.Object(bucket, photo)
        s3_response = s3_object.get()

        stream = io.BytesIO(s3_response['Body'].read())
        image = Image.open(stream)

        image_type = Image.MIME[image.format]

        if (image_type == "image/jpeg" or image_type == "image/png") is False:
            logger.error("Invalid image type for %s", photo)
            raise ValueError(
                f"Invalid file format. Supply a jpeg or png format file:
 {photo}")

        ImageDraw.Draw(image)

        # Call DetectCustomLabels.
        response = rek_client.detect_custom_labels(
            Image={'S3Object': {'Bucket': bucket, 'Name': photo}},
            MinConfidence=min_confidence,
            ProjectVersionArn=model)
```

```
        show_image(image, response)
        return len(response['CustomLabels'])

    except ClientError as err:
        logger.error(format(err))
        raise


def show_image(image, response):
    """
    Displays the analyzed image and overlays analysis results
    :param image: The analyzed image
    :param response: the response from DetectCustomLabels
    """
    try:
        font_size = 40
        line_width = 5

        img_width, img_height = image.size
        draw = ImageDraw.Draw(image)

        # Calculate and display bounding boxes for each detected custom label.
        image_level_label_height = 0

        for custom_label in response['CustomLabels']:
            confidence = int(round(custom_label['Confidence'], 0))
            label_text = f"{custom_label['Name']}:{confidence}%"
            fnt = ImageFont.truetype('Tahoma.ttf', font_size)
            text_left, text_top, text_right, text_bottom = draw.textbbox((0, 0),
 label_text, fnt)
            text_width, text_height = text_right - text_left, text_bottom -
 text_top

            logger.info("Label: %s", custom_label['Name'])
            logger.info("Confidence:  %s", confidence)

            # Draw bounding boxes, if present
            if 'Geometry' in custom_label:
                box = custom_label['Geometry']['BoundingBox']
                left = img_width * box['Left']
                top = img_height * box['Top']
                width = img_width * box['Width']
                height = img_height * box['Height']
```

```
                logger.info("Bounding box")
                logger.info("\tLeft: {0:.0f}".format(left))
                logger.info("\tTop: {0:.0f}".format(top))
                logger.info("\tLabel Width: {0:.0f}".format(width))
                logger.info("\tLabel Height: {0:.0f}".format(height))

                points = (
                    (left, top),
                    (left + width, top),
                    (left + width, top + height),
                    (left, top + height),
                    (left, top))
                # Draw bounding box and label text
                draw.line(points, fill="limegreen", width=line_width)
                draw.rectangle([(left + line_width, top+line_width),
                                (left + text_width + line_width, top +
 line_width + text_height)], fill="black")
                draw.text((left + line_width, top + line_width),
                          label_text, fill="limegreen", font=fnt)

            # draw image-level label text.
            else:
                draw.rectangle([(10, image_level_label_height),
                                (text_width + 10, image_level_label_height
+text_height)], fill="black")
                draw.text((10, image_level_label_height),
                          label_text, fill="limegreen", font=fnt)

                image_level_label_height += text_height

        image.show()

    except Exception as err:
        logger.error(format(err))
        raise


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

    parser.add_argument(
```

```
        "model_arn", help="The ARN of the model that you want to use."
    )

    parser.add_argument(
        "image", help="The path and file name of the image that you want to
 analyze"
    )
    parser.add_argument(
        "--bucket",  help="The bucket that contains the image. If not supplied,
 image is assumed to be a local file.", required=False
    )


def main():

    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()

        label_count = 0
        min_confidence = 50

        session = boto3.Session(profile_name='custom-labels-access')
        rekognition_client = session.client("rekognition")

        if args.bucket is None:
            # Analyze local image.
            label_count = analyze_local_image(rekognition_client,
                                              args.model_arn,
                                              args.image,
                                              min_confidence)
        else:
            # Analyze image in S3 bucket.
            s3_connection = session.resource('s3')
            label_count = analyze_s3_image(rekognition_client,
                                           s3_connection,
                                           args.model_arn,
                                           args.bucket,
                                           args.image,
```

```
                                            min_confidence)

        print(f"Custom labels detected: {label_count}")

    except ClientError as client_err:
        print("A service client error occurred: " +
              format(client_err.response["Error"]["Message"]))

    except ValueError as value_err:
        print("A value error occurred: " + format(value_err))

    except FileNotFoundError as file_error:
        print("File not found error: " + format(file_error))

    except Exception as err:
        print("An error occurred: " + format(err))


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

Java

以下的範例程式碼顯示圖像中找到的邊界框和圖像層級標籤。

若要分析本機圖像，請執行程式並提供以下命令列參數：

- 您要用來分析圖像的模型的 ARN。
- 本機圖像檔案的名稱和位置。

若要分析儲存在 Amazon S3 儲存貯體中的圖像，請執行程式並提供以下命令列參數：

- 您要用來分析圖像的模型的 ARN。
- 您在步驟 4 中使用 Amazon S3 儲存貯體儲存該圖像的名稱和位置。
- 您在步驟 4 中曾用於儲存該圖像的 Amazon S3 儲存貯體。

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

```
package com.amazonaws.samples;

import java.awt.*;
import java.awt.image.BufferedImage;
import java.io.IOException;
import java.util.List;
import javax.imageio.ImageIO;
import javax.swing.*;
import java.io.FileNotFoundException;
import java.awt.font.FontRenderContext;
import java.util.logging.Level;
import java.util.logging.Logger;
import java.io.File;
import java.io.FileInputStream;
import java.io.InputStream;
import java.nio.ByteBuffer;
import java.io.ByteArrayInputStream;
import java.io.ByteArrayOutputStream;

import com.amazonaws.auth.AWSCredentialsProvider;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.rekognition.AmazonRekognition;
import com.amazonaws.services.rekognition.AmazonRekognitionClientBuilder;

import com.amazonaws.services.rekognition.model.BoundingBox;
import com.amazonaws.services.rekognition.model.CustomLabel;
import com.amazonaws.services.rekognition.model.DetectCustomLabelsRequest;
import com.amazonaws.services.rekognition.model.DetectCustomLabelsResult;
import com.amazonaws.services.rekognition.model.Image;
import com.amazonaws.services.rekognition.model.S3Object;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.S3ObjectInputStream;

import com.amazonaws.services.rekognition.model.AmazonRekognitionException;
import com.amazonaws.services.s3.model.AmazonS3Exception;
import com.amazonaws.util.IOUtils;

// Calls DetectCustomLabels and displays a bounding box around each detected
 image.
public class DetectCustomLabels extends JPanel {
```

```
   private transient DetectCustomLabelsResult response;
   private transient Dimension dimension;
   private transient BufferedImage image;

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

   // Finds custom labels in an image stored in an S3 bucket.
   public DetectCustomLabels(AmazonRekognition rekClient,
           AmazonS3 s3client,
           String projectVersionArn,
           String bucket,
           String key,
           Float minConfidence) throws AmazonRekognitionException,
AmazonS3Exception, IOException {

       logger.log(Level.INFO, "Processing S3 bucket: {0} image {1}", new
Object[] { bucket, key });

       // Get image from S3 bucket and create BufferedImage
       com.amazonaws.services.s3.model.S3Object s3object =
s3client.getObject(bucket, key);
       S3ObjectInputStream inputStream = s3object.getObjectContent();
       image = ImageIO.read(inputStream);

       // Set image size
       setWindowDimensions();

       DetectCustomLabelsRequest request = new DetectCustomLabelsRequest()
               .withProjectVersionArn(projectVersionArn)
               .withImage(new Image().withS3Object(new
S3Object().withName(key).withBucket(bucket)))
               .withMinConfidence(minConfidence);

       // Call DetectCustomLabels

       response = rekClient.detectCustomLabels(request);
       logFoundLabels(response.getCustomLabels());
       drawLabels();

   }

   // Finds custom label in a local image file.
   public DetectCustomLabels(AmazonRekognition rekClient,
```

```
            String projectVersionArn,
            String photo,
            Float minConfidence)
            throws IOException, AmazonRekognitionException {

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

        // Get image bytes and buffered image
        ByteBuffer imageBytes;
        try (InputStream inputStream = new FileInputStream(new File(photo))) {
            imageBytes = ByteBuffer.wrap(IOUtils.toByteArray(inputStream));
        }

        // Get image for display
        InputStream imageBytesStream;
        imageBytesStream = new ByteArrayInputStream(imageBytes.array());

        ByteArrayOutputStream baos = new ByteArrayOutputStream();
        image = ImageIO.read(imageBytesStream);
        ImageIO.write(image, "jpg", baos);

        // Set image size
        setWindowDimensions();

        // Analyze image
        DetectCustomLabelsRequest request = new DetectCustomLabelsRequest()
                .withProjectVersionArn(projectVersionArn)
                .withImage(new Image()
                        .withBytes(imageBytes))
                .withMinConfidence(minConfidence);

        response = rekClient.detectCustomLabels(request);

        logFoundLabels(response.getCustomLabels());

        drawLabels();

    }

    // Log the labels found by DetectCustomLabels
    private void logFoundLabels(List<CustomLabel> customLabels) {
        logger.info("Custom labels found");
        if (customLabels.isEmpty()) {
```

```
            logger.log(Level.INFO, "No Custom Labels found. Consider lowering
min confidence.");
        } else {
            for (CustomLabel customLabel : customLabels) {
                logger.log(Level.INFO, " Label: {0} Confidence: {1}",
                        new Object[] { customLabel.getName(),
customLabel.getConfidence() });
            }

        }
    }

    // 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;
        if (image.getWidth() < dimension.width) {
            dimension.width = image.getWidth();
        }
        dimension.height = (int) dimension.getHeight() / 2;

        if (image.getHeight() < dimension.height) {
            dimension.height = image.getHeight();
        }

        setPreferredSize(dimension);

    }

    // Draws the image containing the bounding boxes and labels.
    @Override
    public void paintComponent(Graphics g) {

        Graphics2D g2d = (Graphics2D) g; // Create a Java2D version of g.

        // Draw the image.
        g2d.drawImage(image, 0, 0, dimension.width, dimension.height, this);

    }

    public void drawLabels() {
        // Draws bounding boxes (if present) and label text.
```

```
        int boundingBoxBorderWidth = 5;
        int imageHeight = image.getHeight(this);
        int imageWidth = image.getWidth(this);

        // Set up drawing
        Graphics2D g2d = image.createGraphics();
        g2d.setColor(Color.GREEN);
        g2d.setFont(new Font("Tahoma", Font.PLAIN, 50));
        Font font = g2d.getFont();
        FontRenderContext frc = g2d.getFontRenderContext();
        g2d.setStroke(new BasicStroke(boundingBoxBorderWidth));

        List<CustomLabel> customLabels = response.getCustomLabels();

        int imageLevelLabelHeight = 0;
        for (CustomLabel customLabel : customLabels) {

            String label = customLabel.getName();

            int textWidth = (int) (font.getStringBounds(label, frc).getWidth());
            int textHeight = (int) (font.getStringBounds(label,
frc).getHeight());

            // Draw bounding box, if present
            if (customLabel.getGeometry() != null) {

                BoundingBox box = customLabel.getGeometry().getBoundingBox();
                float left = imageWidth * box.getLeft();
                float top = imageHeight * box.getTop();

                // Draw black rectangle
                g2d.setColor(Color.BLACK);
                g2d.fillRect(Math.round(left + (boundingBoxBorderWidth)),
Math.round(top + (boundingBoxBorderWidth)),
                        textWidth + boundingBoxBorderWidth, textHeight +
boundingBoxBorderWidth);

                // Write label onto black rectangle
                g2d.setColor(Color.GREEN);
                g2d.drawString(label, left + boundingBoxBorderWidth, (top +
textHeight));

                // Draw bounding box around label location
```

```
                g2d.drawRect(Math.round(left), Math.round(top),
 Math.round((imageWidth * box.getWidth())),
                        Math.round((imageHeight * box.getHeight())));
            }
            // Draw image level labels.
            else {
                // Draw black rectangle
                g2d.setColor(Color.BLACK);
                g2d.fillRect(10, 10 + imageLevelLabelHeight, textWidth,
 textHeight);
                g2d.setColor(Color.GREEN);
                g2d.drawString(label, 10, textHeight + imageLevelLabelHeight);

                imageLevelLabelHeight += textHeight;
            }

        }
        g2d.dispose();

    }

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

        String photo = null;
        String bucket = null;
        String projectVersionArn = null;
        float minConfidence = 50;

        final String USAGE = "\n" + "Usage: " + "<model_arn> <image> <bucket>\n
\n" + "Where:\n"
                + "   model_arn - The ARN of the model that you want to use. \n
\n"
                + "   image - The location of the image on your local file
 system or within an S3 bucket.\n\n"
                + "   bucket - The S3 bucket that contains the image. Don't
 specify if image is local.\n\n";

        // Collect the arguments. If 3 arguments are present, the image is
 assumed to be
        // in an S3 bucket.

        if (args.length < 2 || args.length > 3) {
            System.out.println(USAGE);
            System.exit(1);
```

```
        }

        projectVersionArn = args[0];
        photo = args[1];

        if (args.length == 3) {
            bucket = args[2];
        }

        DetectCustomLabels panel = null;

        try {

            AWSCredentialsProvider provider =new
ProfileCredentialsProvider("custom-labels-access");

            AmazonRekognition rekClient =
AmazonRekognitionClientBuilder.standard()
                    .withCredentials(provider)
                    .withRegion(Regions.US_WEST_2)
                    .build();


            AmazonS3 s3client = AmazonS3ClientBuilder.standard()
            .withCredentials(provider)
            .withRegion(Regions.US_WEST_2)
            .build();

            // Create frame and panel.
            JFrame frame = new JFrame("Custom Labels");
            frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

            if (args.length == 2) {
                // Analyze local image
                panel = new DetectCustomLabels(rekClient, projectVersionArn,
photo, minConfidence);
            } else {
                // Analyze image in S3 bucket
                panel = new DetectCustomLabels(rekClient, s3client,
projectVersionArn, bucket, photo, minConfidence);
            }

            frame.setContentPane(panel);
            frame.pack();
```

```
            frame.setVisible(true);

        } catch (AmazonRekognitionException rekError) {
            String errorMessage = "Rekognition client error: " +
 rekError.getMessage();
            logger.log(Level.SEVERE, errorMessage);
            System.out.println(errorMessage);
            System.exit(1);
        } catch (FileNotFoundException fileError) {
            String errorMessage = "File not found: " + photo;
            logger.log(Level.SEVERE, errorMessage);
            System.out.println(errorMessage);
            System.exit(1);
        } catch (IOException fileError) {
            String errorMessage = "Input output exception: " +
 fileError.getMessage();
            logger.log(Level.SEVERE, errorMessage);
            System.out.println(errorMessage);
            System.exit(1);
        } catch (AmazonS3Exception s3Error) {
            String errorMessage = "S3 error: " + s3Error.getErrorMessage();
            logger.log(Level.SEVERE, errorMessage);
            System.out.println(errorMessage);
            System.exit(1);
        }

    }
}
```

Java V2

以下的範例程式碼顯示圖像中找到的邊界框和圖像層級標籤。

若要分析本機圖像，請執行程式並提供以下命令列參數：

- projectVersionArn – 您要用來分析圖像的模型的 ARN。
- photo – 本機圖像檔案的名稱和位置。

若要分析儲存在 S3 儲存貯體中的圖像，請執行程式並提供以下命令列參數：

- 您要用來分析圖像的模型的 ARN。
- 您在步驟 4 中使用 S3 儲存貯體儲存該圖像的名稱和位置。

- 您在步驟 4 中曾用於儲存該圖像的 Amazon S3 儲存貯體。

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

package com.example.rekognition;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.core.ResponseBytes;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.core.sync.ResponseTransformer;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.S3Object;
import software.amazon.awssdk.services.rekognition.model.Image;
import
 software.amazon.awssdk.services.rekognition.model.DetectCustomLabelsRequest;
import
 software.amazon.awssdk.services.rekognition.model.DetectCustomLabelsResponse;
import software.amazon.awssdk.services.rekognition.model.CustomLabel;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import software.amazon.awssdk.services.rekognition.model.BoundingBox;

import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.GetObjectResponse;
import software.amazon.awssdk.services.s3.model.NoSuchBucketException;
import software.amazon.awssdk.services.s3.model.NoSuchKeyException;

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.util.List;

import java.awt.*;
import java.awt.font.FontRenderContext;
import java.awt.image.BufferedImage;
import javax.imageio.ImageIO;
```

```
import javax.swing.*;

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

// Calls DetectCustomLabels on an image. Displays bounding boxes or
// image level labels found in the image.
public class ShowCustomLabels extends JPanel {

    private transient BufferedImage image;
    private transient DetectCustomLabelsResponse response;
    private transient Dimension dimension;
    public static final Logger logger =
 Logger.getLogger(ShowCustomLabels.class.getName());

    // Finds custom labels in an image stored in an S3 bucket.
    public ShowCustomLabels(RekognitionClient rekClient,
            S3Client s3client,
            String projectVersionArn,
            String bucket,
            String key,
            Float minConfidence) throws RekognitionException,
 NoSuchBucketException, NoSuchKeyException, IOException {

        logger.log(Level.INFO, "Processing S3 bucket: {0} image {1}", new
 Object[] { bucket, key });
        // Get image from S3 bucket and create BufferedImage
        GetObjectRequest requestObject =
 GetObjectRequest.builder().bucket(bucket).key(key).build();
        ResponseBytes<GetObjectResponse> result =
 s3client.getObject(requestObject, ResponseTransformer.toBytes());
        ByteArrayInputStream bis = new
 ByteArrayInputStream(result.asByteArray());
        image = ImageIO.read(bis);

        // Set image size
        setWindowDimensions();

        // Construct request parameter for DetectCustomLabels
        S3Object s3Object = S3Object.builder().bucket(bucket).name(key).build();

        Image s3Image = Image.builder().s3Object(s3Object).build();
```

```
        DetectCustomLabelsRequest request =
DetectCustomLabelsRequest.builder().image(s3Image)

.projectVersionArn(projectVersionArn).minConfidence(minConfidence).build();

        response = rekClient.detectCustomLabels(request);
        logFoundLabels(response.customLabels());
        drawLabels();

    }

    // Finds custom label in a local image file.
    public ShowCustomLabels(RekognitionClient rekClient,
            String projectVersionArn,
            String photo,
            Float minConfidence)
            throws IOException, RekognitionException {

        logger.log(Level.INFO, "Processing local file: {0}", photo);
        // Get image bytes and buffered image
        InputStream sourceStream = new FileInputStream(new File(photo));
        SdkBytes imageBytes = SdkBytes.fromInputStream(sourceStream);
        ByteArrayInputStream inputStream = new
ByteArrayInputStream(imageBytes.asByteArray());
        image = ImageIO.read(inputStream);

        setWindowDimensions();

        // Construct request parameter for DetectCustomLabels
        Image localImageBytes = Image.builder().bytes(imageBytes).build();

        DetectCustomLabelsRequest request =
DetectCustomLabelsRequest.builder().image(localImageBytes)

.projectVersionArn(projectVersionArn).minConfidence(minConfidence).build();

        response = rekClient.detectCustomLabels(request);

        logFoundLabels(response.customLabels());
        drawLabels();

    }

    // 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;
        if (image.getWidth() < dimension.width) {
            dimension.width = image.getWidth();
        }
        dimension.height = (int) dimension.getHeight() / 2;

        if (image.getHeight() < dimension.height) {
            dimension.height = image.getHeight();
        }

        setPreferredSize(dimension);

    }

    // Draws bounding boxes (if present) and label text.
    public void drawLabels() {

        int boundingBoxBorderWidth = 5;
        int imageHeight = image.getHeight(this);
        int imageWidth = image.getWidth(this);

        // Set up drawing
        Graphics2D g2d = image.createGraphics();
        g2d.setColor(Color.GREEN);
        g2d.setFont(new Font("Tahoma", Font.PLAIN, 50));
        Font font = g2d.getFont();
        FontRenderContext frc = g2d.getFontRenderContext();
        g2d.setStroke(new BasicStroke(boundingBoxBorderWidth));

        List<CustomLabel> customLabels = response.customLabels();

        int imageLevelLabelHeight = 0;
        for (CustomLabel customLabel : customLabels) {

            String label = customLabel.name();

            int textWidth = (int) (font.getStringBounds(label, frc).getWidth());
            int textHeight = (int) (font.getStringBounds(label,
frc).getHeight());

            // Draw bounding box, if present
```

```
            if (customLabel.geometry() != null) {

                BoundingBox box = customLabel.geometry().boundingBox();
                float left = imageWidth * box.left();
                float top = imageHeight * box.top();

                // Draw black rectangle
                g2d.setColor(Color.BLACK);
                g2d.fillRect(Math.round(left + (boundingBoxBorderWidth)),
Math.round(top + (boundingBoxBorderWidth)),
                        textWidth + boundingBoxBorderWidth, textHeight +
boundingBoxBorderWidth);

                // Write label onto black rectangle
                g2d.setColor(Color.GREEN);
                g2d.drawString(label, left + boundingBoxBorderWidth, (top +
textHeight));

                // Draw bounding box around label location
                g2d.drawRect(Math.round(left), Math.round(top),
Math.round((imageWidth * box.width())),
                        Math.round((imageHeight * box.height())));
            }
            // Draw image level labels.
            else {
                // Draw black rectangle
                g2d.setColor(Color.BLACK);
                g2d.fillRect(10, 10 + imageLevelLabelHeight, textWidth,
textHeight);
                g2d.setColor(Color.GREEN);
                g2d.drawString(label, 10, textHeight + imageLevelLabelHeight);

                imageLevelLabelHeight += textHeight;
            }

        }
        g2d.dispose();

    }

    // Log the labels found by DetectCustomLabels
    private void logFoundLabels(List<CustomLabel> customLabels) {
        logger.info("Custom labels found:");
        if (customLabels.isEmpty()) {
```

```
            logger.log(Level.INFO, "No Custom Labels found. Consider lowering
 min confidence.");

        }
        else {
        for (CustomLabel customLabel : customLabels) {
            logger.log(Level.INFO, " Label: {0} Confidence: {1}",
                    new Object[] { customLabel.name(),
 customLabel.confidence() } );
            }
        }
    }

    // Draws the image containing the bounding boxes and labels.
    @Override
    public void paintComponent(Graphics g) {

        Graphics2D g2d = (Graphics2D) g; // Create a Java2D version of g.

        // Draw the image.
        g2d.drawImage(image, 0, 0, dimension.width, dimension.height, this);

    }

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

        String photo = null;
        String bucket = null;
        String projectVersionArn = null;

        final String USAGE = "\n" + "Usage: " + "<model_arn> <image> <bucket>\n
\n" + "Where:\n"
                + "   model_arn - The ARN of the model that you want to use. \n
\n"
                + "   image - The location of the image on your local file
 system or within an S3 bucket.\n\n"
                + "   bucket - The S3 bucket that contains the image. Don't
 specify if image is local.\n\n";

        // Collect the arguments. If 3 arguments are present, the image is
 assumed to be
        // in an S3 bucket.

        if (args.length < 2 || args.length > 3) {
```

```
            System.out.println(USAGE);
            System.exit(1);
        }

        projectVersionArn = args[0];
        photo = args[1];

        if (args.length == 3) {
            bucket = args[2];
        }

        float minConfidence = 50;

        ShowCustomLabels panel = null;

        try {
            // Get the Rekognition client


            // Get the Rekognition client.
            RekognitionClient rekClient = RekognitionClient.builder()
            .credentialsProvider(ProfileCredentialsProvider.create("custom-
labels-access"))
            .region(Region.US_WEST_2)
            .build();

            S3Client s3Client = S3Client.builder()
            .credentialsProvider(ProfileCredentialsProvider.create("custom-
labels-access"))
            .region(Region.US_WEST_2)
            .build();


            // Create frame and panel.
            JFrame frame = new JFrame("Custom Labels");
            frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

            if (args.length == 2) {
                // Analyze local image
                panel = new ShowCustomLabels(rekClient, projectVersionArn,
 photo, minConfidence);
            } else {
                // Analyze image in S3 bucket
```

```
                panel = new ShowCustomLabels(rekClient, s3Client,
 projectVersionArn, bucket, photo, minConfidence);
            }

            frame.setContentPane(panel);
            frame.pack();
            frame.setVisible(true);

        } catch (RekognitionException rekError) {

            String errorMessage = "Rekognition client error: " +
 rekError.getMessage();
            logger.log(Level.SEVERE, errorMessage);
            System.out.println(errorMessage);
            System.exit(1);
        } catch (FileNotFoundException fileError) {
            String errorMessage = "File not found: " + photo;
            logger.log(Level.SEVERE, errorMessage);
            System.out.println(errorMessage);
            System.exit(1);
        } catch (IOException fileError) {
            String errorMessage = "Input output exception: " +
 fileError.getMessage();
            logger.log(Level.SEVERE, errorMessage);
            System.out.println(errorMessage);
            System.exit(1);
        } catch (NoSuchKeyException bucketError) {
            String errorMessage = String.format("Image not found: %s in bucket
 %s.", photo, bucket);
            logger.log(Level.SEVERE, errorMessage);
            System.out.println(errorMessage);
            System.exit(1);
        } catch (NoSuchBucketException bucketError) {
            String errorMessage = "Bucket not found: " + bucket;
            logger.log(Level.SEVERE, errorMessage);
            System.out.println(errorMessage);
            System.exit(1);
        }

    }
}
```

DetectCustomLabels 操作請求

在 DetectCustomLabels 操作中，提供 Base64 編碼位元組陣列的輸入映像或是儲存於 Amazon S3 儲存貯體中的映像。以下範例中的 JSON 請求顯示了從 Amazon S3 儲存貯體載入的圖像。

```
{
    "ProjectVersionArn": "string",
     "Image":{
        "S3Object":{
            "Bucket":"string",
            "Name":"string",
            "Version":"string"
         }
    },
    "MinConfidence": 90,
    "MaxLabels": 10,
}
```

DetectCustomLabels 作業回應

以下來自 DetectCustomLabels 操作的 JSON 回應顯示了下圖中偵測到的自訂標籤。

```
{
    "CustomLabels": [
        {
            "Name": "MyLogo",
            "Confidence": 77.7729721069336,
            "Geometry": {
                "BoundingBox": {
                    "Width": 0.198987677693367,
                    "Height": 0.31296101212501526,
                    "Left": 0.07924537360668182,
                    "Top": 0.4037395715713501
                }
            }
        }
    ]
}
```

管理 Amazon Rekognition 自訂標籤資源

本節提供用於訓練和使用 Amazon Rekognition 自訂標籤模型的工作流程概觀。此外，還包括使用 AWS SDK 訓練和使用模型的概觀資訊。

管理 Amazon Rekognition 自訂標籤專案

在 Amazon Rekognition 自訂標籤中，您可以使用專案來管理針對特定使用案例建立的模型。專案會管理資料集、模型訓練、模型版本、模型評估，以及專案模型的執行。

主題

- 刪除 Amazon Rekognition 自訂標籤專案
- 描述一個項目（SDK）
- 建立專案AWS CloudFormation

刪除 Amazon Rekognition 自訂標籤專案

您可以使用 Amazon Rekognition 主控台或呼叫 DeleteProjectAPI 來刪除專案。若要刪除專案，您必須先刪除每個相關聯模型。刪除的專案或模型無法取消刪除。

主題

- 刪除 Amazon Rekognition 自訂標籤專案 (主控台)
- 刪除 Amazon Rekognition 自訂標籤專案 (SDK)

刪除 Amazon Rekognition 自訂標籤專案 (主控台)

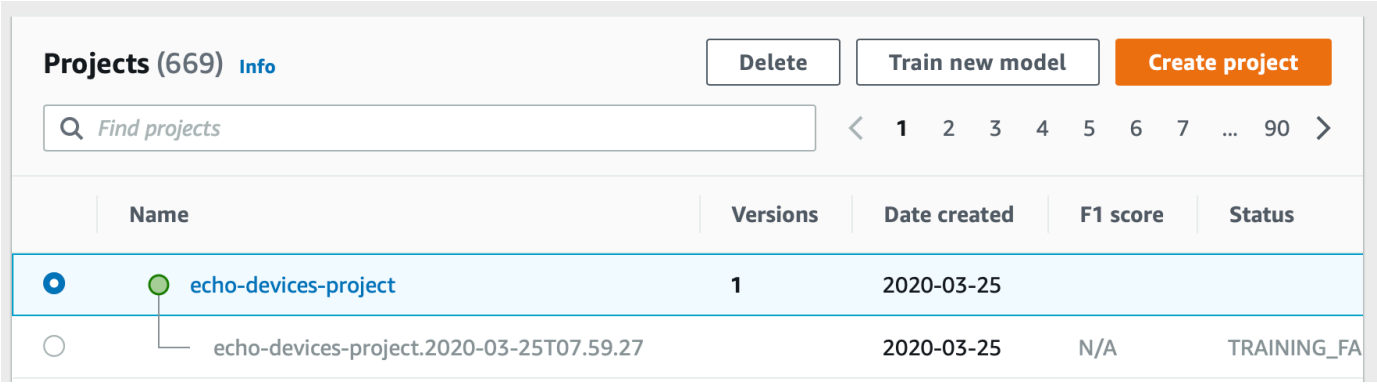
您可以從專案頁面刪除專案，也可以從專案的詳細資訊頁面刪除專案。下列程序顯示如何使用專案 (專案) 頁面來刪除專案。

Amazon Rekognition 自訂標籤主控台會在專案刪除期間為您刪除相關的模型和資料集。如果專案的任何模型正在執行或訓練，則無法刪除專案。若要停止執行中模型，請參閱停止 Amazon Rekognition 自訂標籤模型 (SDK)。如果模型正在訓練，請等到完成後再刪除專案。

若要刪除專案 (主控台)

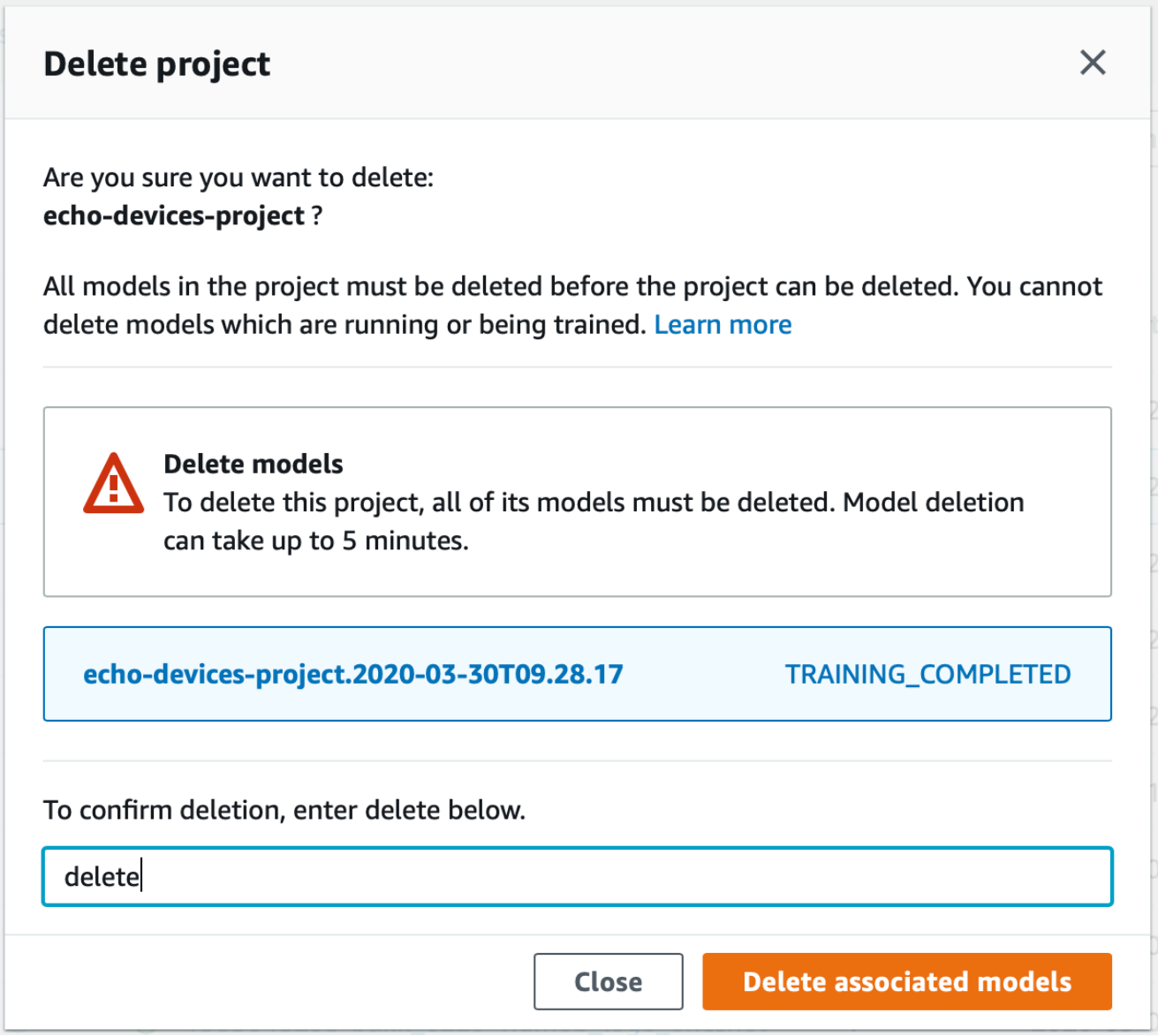
1. 開啟亞馬遜重新認知主控台，網址為 https://console.aws.amazon.com/rekognition/。

2. 選擇「使用自訂標籤」。
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 (專案) 頁面上，為您要刪除專案選擇圓形按鈕。



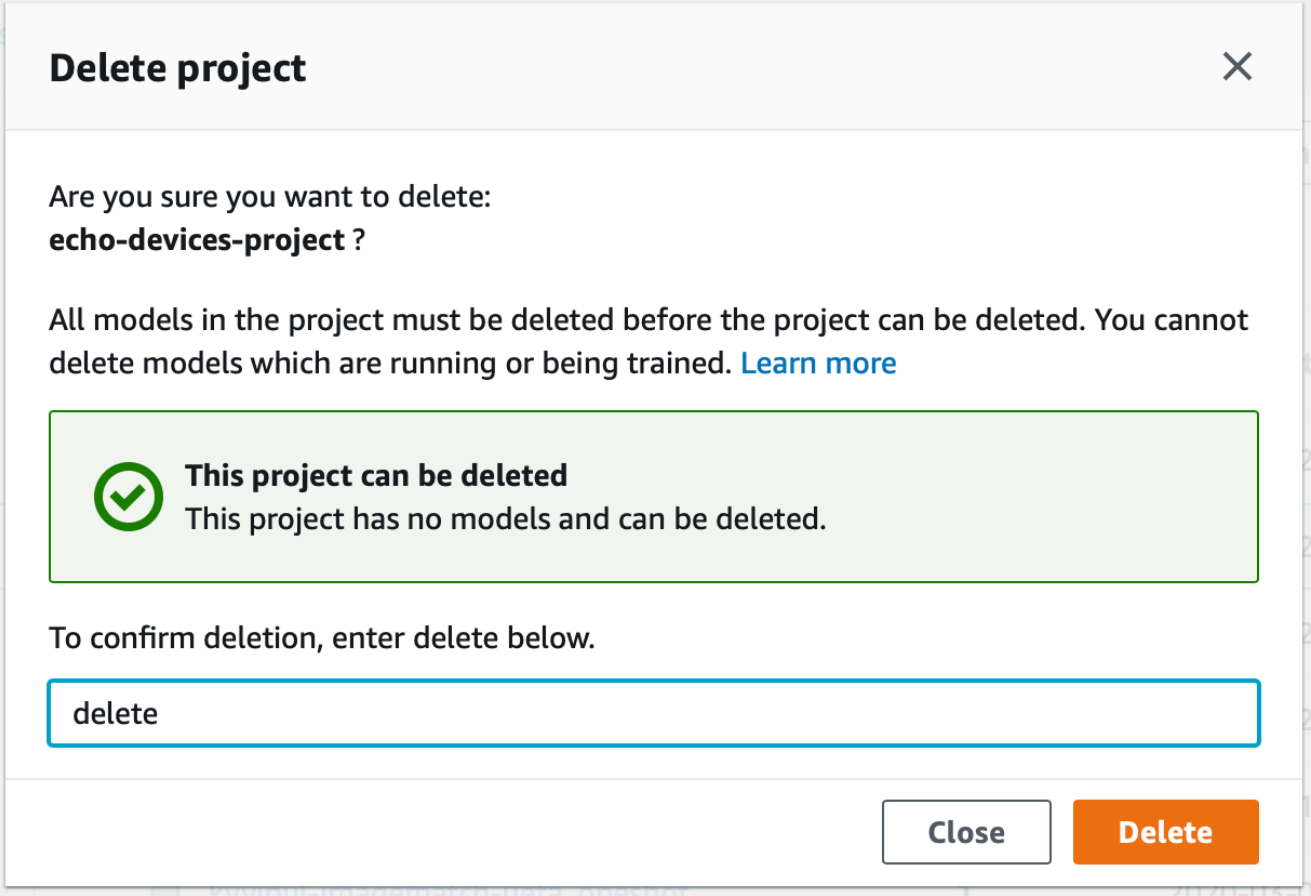
6. 在頁面頂端，選擇 Delete (刪除)。螢幕上將顯示「刪除專案」對話方塊。
7. 如果專案沒有關聯的模型：
   a. 輸入刪除以刪除專案。
   b. 選擇「刪除」以刪除專案。
8. 如果專案具有相關聯的模型或資料集：
   a. 輸入 delete 以確認您要刪除模型和資料集。
   b. 視模型是否具有資料集、模型或兩者而定，選擇「刪除關聯的模型」或「刪除關聯的資料集」或「刪除關聯的資料集和模型」。模型刪除可能需要一段時間才能完成。

> **Note**
>
> 主控台無法刪除正在訓練或執行中的模型。停止列出的任何執行中模型後再試一次，然後等到列為訓練的模型完成。
> 如果您在刪除模型期間關閉對話方塊，模型仍會被刪除。稍後，您可以通過重複此過程來刪除該項目。



c. 輸入 delete 以確認您要刪除專案。

d. 選擇「刪除」以刪除專案。



刪除 Amazon Rekognition 自訂標籤專案 (SDK)

若要刪除 Amazon Rekognition 自訂標籤專案，即可呼叫DeleteProject並提供您要刪除專案的 Amazon Rekognition 自訂標籤專案。若要取得AWS帳戶中專案的 ARN，請撥打DescribeProjects。響應包括對ProjectDescription象的數組。該項目 ARN 是字`ProjectArn`段。您可以使用專案名稱來識別專案的 ARN。例如：`arn:aws:rekognition:us-east-1:123456789010:project/`*`project name`*`/1234567890123`。

在刪除專案之前，您必須先刪除專案中的所有模型和資料集。如需詳細資訊，請參閱 刪除 Amazon Rekognition 自訂標籤模型 (SDK) 及 刪除資料集。

專案可能需要一點時間來刪除專案。在此期間，專案的狀態為`DELETING`。如果後續呼叫DescribeProjects未包含您刪除的專案，則會刪除專案。

若要刪除專案 (SDK)

1. 若您尚未這樣做，請安裝AWS CLI並設定和AWS SDK。如需詳細資訊，請參閱步驟 4：設定 AWS CLI 和 AWS SDKs。

2. 使用下列程式碼來刪除專案。

AWS CLI

project-arn將的值變更為您要刪除專案的名稱。

```
aws rekognition delete-project --project-arn project_arn \
  --profile custom-labels-access
```

Python

使用下列程式碼。提供以下命令行參數：

- project_arn— 您要刪除專案的 ARN。

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

"""
Purpose
Amazon Rekognition Custom Labels project example used in the service
 documentation:
https://docs.aws.amazon.com/rekognition/latest/customlabels-dg/mp-delete-
project.html
Shows how to delete an existing Amazon Rekognition Custom Labels project.
You must first delete any models and datasets that belong to the project.
"""

import argparse
import logging
import time
import boto3


from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)


def find_forward_slash(input_string, n):
    """
```

```
    Returns the location of '/' after n number of occurences.
    :param input_string: The string you want to search
    : n: the occurence that you want to find.
    """
    position = input_string.find('/')
    while position >= 0 and n > 1:
        position = input_string.find('/', position + 1)
        n -= 1
    return position


def delete_project(rek_client, project_arn):
    """
    Deletes an Amazon Rekognition Custom Labels project.
    :param rek_client: The Amazon Rekognition Custom Labels Boto3 client.
    :param project_arn: The ARN of the project that you want to delete.
    """

    try:
        # Delete the project
        logger.info("Deleting project: %s", project_arn)

        response = rek_client.delete_project(ProjectArn=project_arn)

        logger.info("project status: %s",response['Status'])

        deleted = False

        logger.info("waiting for project deletion: %s", project_arn)

        # Get the project name
        start = find_forward_slash(project_arn, 1) + 1
        end = find_forward_slash(project_arn, 2)
        project_name = project_arn[start:end]

        project_names = [project_name]

        while deleted is False:

            project_descriptions = rek_client.describe_projects(
                ProjectNames=project_names)['ProjectDescriptions']

            if len(project_descriptions) == 0:
                deleted = True
```

```
            else:
                time.sleep(5)

        logger.info("project deleted: %s",project_arn)

        return True

    except ClientError as err:
        logger.exception(
            "Couldn't delete project - %s: %s",
            project_arn, err.response['Error']['Message'])
        raise


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

    parser.add_argument(
        "project_arn", help="The ARN of the project that you want to delete."
    )


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()

        print(f"Deleting project: {args.project_arn}")

        # Delete the project.
        session = boto3.Session(profile_name='custom-labels-access')
        rekognition_client = session.client("rekognition")
```

```
        delete_project(rekognition_client,
                       args.project_arn)

        print(f"Finished deleting project: {args.project_arn}")

    except ClientError as err:
        error_message = f"Problem deleting project: {err}"
        logger.exception(error_message)
        print(error_message)


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

Java V2

使用下列程式碼。提供以下命令行參數：

- project_arn— 您要刪除專案的 ARN。

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

package com.example.rekognition;

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

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.DeleteProjectRequest;
import software.amazon.awssdk.services.rekognition.model.DeleteProjectResponse;
import
 software.amazon.awssdk.services.rekognition.model.DescribeProjectsRequest;
import
 software.amazon.awssdk.services.rekognition.model.DescribeProjectsResponse;
import software.amazon.awssdk.services.rekognition.model.ProjectDescription;
```

```
import software.amazon.awssdk.services.rekognition.model.RekognitionException;

public class DeleteProject {

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

    public static void deleteMyProject(RekognitionClient rekClient, String
 projectArn) throws InterruptedException {

        try {

            logger.log(Level.INFO, "Deleting project: {0}", projectArn);

            // Delete the project

            DeleteProjectRequest deleteProjectRequest =
 DeleteProjectRequest.builder().projectArn(projectArn).build();
            DeleteProjectResponse response =
 rekClient.deleteProject(deleteProjectRequest);

            logger.log(Level.INFO, "Status: {0}", response.status());

            // Wait until deletion finishes

            Boolean deleted = false;

            do {

        DescribeProjectsRequest describeProjectsRequest =
 DescribeProjectsRequest.builder().build();
                    DescribeProjectsResponse describeResponse =
 rekClient.describeProjects(describeProjectsRequest);
                    List<ProjectDescription> projectDescriptions =
 describeResponse.projectDescriptions();

                    deleted = true;

                    for (ProjectDescription projectDescription :
 projectDescriptions) {

                        if (Objects.equals(projectDescription.projectArn(),
 projectArn)) {
                            deleted = false;
```

```
                            logger.log(Level.INFO, "Not deleted: {0}",
 projectDescription.projectArn());
                            Thread.sleep(5000);
                            break;
                        }
                    }

            } while (Boolean.FALSE.equals(deleted));

            logger.log(Level.INFO, "Project deleted: {0} ", projectArn);

        } catch (

        RekognitionException e) {
            logger.log(Level.SEVERE, "Client error occurred: {0}",
 e.getMessage());
            throw e;
        }

    }

    public static void main(String[] args) {

        final String USAGE = "\n" + "Usage: " + "<project_arn>\n\n" + "Where:\n"
             + "   project_arn - The ARN of the project that you want to delete.
\n\n";

        if (args.length != 1) {
             System.out.println(USAGE);
          System.exit(1);
        }

        String projectArn = args[0];

        try {

            RekognitionClient rekClient = RekognitionClient.builder()
                .region(Region.US_WEST_2)
                .credentialsProvider(ProfileCredentialsProvider.create("custom-
labels-access"))
                .build();

            // Delete the project.
            deleteMyProject(rekClient, projectArn);
```

```
            System.out.println(String.format("Project deleted: %s",
 projectArn));

            rekClient.close();

        } catch (RekognitionException rekError) {
            logger.log(Level.SEVERE, "Rekognition client error: {0}",
 rekError.getMessage());
            System.exit(1);
        }

        catch (InterruptedException intError) {
            logger.log(Level.SEVERE, "Exception while sleeping: {0}",
 intError.getMessage());
            System.exit(1);
        }
    }

}
```

描述一個項目（SDK）

您可以使用`DescribeProjects` API 來取得專案的相關資訊。

若要描述專案 (SDK)

1. 若您尚未這樣做，請安裝AWS CLI並設定和AWS SDK。如需詳細資訊，請參閱步驟 4：設定 AWS CLI 和 AWS SDKs。
2. 使用下列範例程式碼來描述專案。以您要描述專案的名稱取`project_name`代。若您沒有指定`--project-names`，就會傳回所有專案的描述。

   AWS CLI

   ```
   aws rekognition describe-projects --project-names project_name \
     --profile custom-labels-access
   ```

   Python

   使用下列程式碼。提供以下命令行參數：

- 專案名稱 — 您要描述專案的名稱。若您沒有指定名稱，就會傳回所有專案的描述。

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

"""
Purpose
Shows how to describe an Amazon Rekognition Custom Labels project.
"""
import argparse
import logging
import json
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)


def display_project_info(project):
    """
    Displays information about a Custom Labels project.
    :param project: The project that you want to display information about.
    """
    print(f"Arn: {project['ProjectArn']}")
    print(f"Status: {project['Status']}")

    if len(project['Datasets']) == 0:
        print("Datasets: None")
    else:
        print("Datasets:")

    for dataset in project['Datasets']:
        print(f"\tCreated: {str(dataset['CreationTimestamp'])}")
        print(f"\tType: {dataset['DatasetType']}")
        print(f"\tARN: {dataset['DatasetArn']}")
        print(f"\tStatus: {dataset['Status']}")
        print(f"\tStatus message: {dataset['StatusMessage']}")
        print(f"\tStatus code: {dataset['StatusMessageCode']}")
        print()
    print()
```

```
def describe_projects(rek_client, project_name):
    """
    Describes an Amazon Rekognition Custom Labels project, or all projects.
    :param rek_client: The Amazon Rekognition Custom Labels Boto3 client.
    :param project_name: The project you want to describe. Pass None to describe
 all projects.
    """

    try:
        # Describe the project
        if project_name is None:
            logger.info("Describing all projects.")
        else:
            logger.info("Describing project: %s.",project_name)

        if project_name is None:
            response = rek_client.describe_projects()
        else:
            project_names = json.loads('["' + project_name + '"]')
            response = rek_client.describe_projects(ProjectNames=project_names)

        print('Projects\n--------')
        if len(response['ProjectDescriptions']) == 0:
            print("Project(s) not found.")
        else:
            for project in response['ProjectDescriptions']:
                display_project_info(project)

        logger.info("Finished project description.")

    except ClientError as err:
        logger.exception(
            "Couldn't describe project - %s: %s",
            project_name,err.response['Error']['Message'] )
        raise


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

    parser.add_argument(
```

```
        "--project_name",  help="The name of the project that you want to
 describe.", 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()

        print(f"Describing projects: {args.project_name}")

        # Describe the project.
        session = boto3.Session(profile_name='custom-labels-access')
        rekognition_client = session.client("rekognition")

        describe_projects(rekognition_client,
                          args.project_name)

        if args.project_name is None:
            print("Finished describing all projects.")
        else:
            print("Finished describing project %s.", args.project_name)

    except ClientError as err:
        error_message = f"Problem describing project: {err}"
        logger.exception(error_message)
        print(error_message)


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

Java V2

使用下列程式碼。提供以下命令行參數：

- project_name— 您要描述專案的 ARN。若您沒有指定名稱，就會傳回所有專案的描述。

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

package com.example.rekognition;

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

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.DatasetMetadata;
import
 software.amazon.awssdk.services.rekognition.model.DescribeProjectsRequest;
import
 software.amazon.awssdk.services.rekognition.model.DescribeProjectsResponse;
import software.amazon.awssdk.services.rekognition.model.ProjectDescription;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;

public class DescribeProjects {

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

    public static void describeMyProjects(RekognitionClient rekClient, String
 projectName) {

        DescribeProjectsRequest descProjects = null;

        // If a single project name is supplied, build projectNames argument

        List<String> projectNames = new ArrayList<String>();
```

```
        if (projectName == null) {
            descProjects = DescribeProjectsRequest.builder().build();
        } else {
            projectNames.add(projectName);
            descProjects =
 DescribeProjectsRequest.builder().projectNames(projectNames).build();
        }

        // Display useful information for each project.

        DescribeProjectsResponse resp =
 rekClient.describeProjects(descProjects);

        for (ProjectDescription projectDescription : resp.projectDescriptions())
 {

            System.out.println("ARN: " + projectDescription.projectArn());
            System.out.println("Status: " +
 projectDescription.statusAsString());
            if (projectDescription.hasDatasets()) {
                for (DatasetMetadata datasetDescription :
 projectDescription.datasets()) {
                    System.out.println("\tdataset Type: " +
 datasetDescription.datasetTypeAsString());
                    System.out.println("\tdataset ARN: " +
 datasetDescription.datasetArn());
                    System.out.println("\tdataset Status: " +
 datasetDescription.statusAsString());
                }
            }
            System.out.println();
        }

    }

    public static void main(String[] args) {

        String projectArn = null;

        // Get command line arguments

        final String USAGE = "\n" + "Usage: " + "<project_name>\n\n" + "Where:
\n"
```

```
                + "   project_name - (Optional) The name of the project that you
 want to describe. If not specified, all projects "
                + "are described.\n\n";

        if (args.length > 1) {
            System.out.println(USAGE);
            System.exit(1);
        }

        if (args.length == 1) {
            projectArn = args[0];
        }

        try {

            // Get the Rekognition client
            RekognitionClient rekClient = RekognitionClient.builder()
                .credentialsProvider(ProfileCredentialsProvider.create("custom-
labels-access"))
                .region(Region.US_WEST_2)
                .build();

            // Describe projects

            describeMyProjects(rekClient, projectArn);

            rekClient.close();

        } catch (RekognitionException rekError) {
            logger.log(Level.SEVERE, "Rekognition client error: {0}",
 rekError.getMessage());
            System.exit(1);
        }

    }

}
```

建立專案AWS CloudFormation

Amazon Rekognition 自訂標籤已整合AWS CloudFormation，這項服務可協助您建立AWS資源的模型和設定，以減少建立和基礎設施的時間。您建立一個描述所有所有所有所有所有所有所AWS CloudFormation需之AWS資源的範本，而負責為您佈建與設定這些資源。

您可以用AWS CloudFormation來佈建和設定 Amazon Rekognition 自訂標籤專案。

當您使用時AWS CloudFormation，您可以重複使用您的範本，重複、一致的設定您的 Amazon Rekognition 自訂標籤專案。只需描述專案一次，即可在多個AWS帳戶與區域內重複佈建相同專案。

Amazon Rekognition 自訂標籤和AWS CloudFormation範本

若要佈建和設定 Amazon Rekognition 自訂標籤和相關服務的專案，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 Amazon Rekognition 自訂標籤專案的參考資訊 (包括 JSON 和 YAML 範本範例)，請參閱 Rekognition 自訂標籤專案 (包括 JSON 和 YAML 範本範例)。

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》
- AWS CloudFormation API 參考
- 《AWS CloudFormation 命令列介面使用者指南》

管理資料集

資料集包含您用來訓練或測試模型的影像和指派的標籤。本節主題說明如何使用 Amazon Rekognition 自訂標籤主控台和AWS開發套件管理資料集的資料集。

主題

- 將資料集新增至專案
- 將更多影像新增至資料集

- 使用現有資料集 (SDK) 建立資料集
- 描述資料集 (SDK)
- 列出資料集項目 (SDK)
- 發佈訓練資料集 (SDK)
- 刪除資料集

將資料集新增至專案

您可以將訓練資料集或測試資料集新增至現有專案。如果您要取代現有的資料集，請先刪除現有的資料集。如需詳細資訊，請參閱刪除資料集。然後，添加新的數據集。

主題

- 將資料集新增至專案 (主控台)
- 將資料集新增至專案 (SDK)

將資料集新增至專案 (主控台)

您可以使用 Amazon Rekognition 自訂標籤主控台，將訓練或測試資料集新增至專案。

將資料集新增至專案

1. 開啟亞馬遜重新認知主控台，網址為 https://console.aws.amazon.com/rekognition/。
2. 在左窗格中選擇使用自訂標籤。顯示 Amazon Rekognition 自訂標籤登陸頁面。
3. 在左側導覽窗格中選擇 Projects (專案)。將顯示「專案」檢視。
4. 選擇要新增資料集的專案。
5. 在左側導覽窗格的專案名稱下，選擇資料集。
6. 如果專案沒有現有的資料集，則會顯示「建立資料集」頁面。請執行下列動作：
    a. 在 [建立資料集] 頁面上，輸入影像來源資訊。如需詳細資訊，請參閱the section called “建立包含影像的資料集”。
    b. 選擇 [建立資料集] 以建立資料集。
7. 如果專案具有現有的資料集 (訓練或測試)，則會顯示專案詳細資料頁面。請執行下列動作：
    a. 在專案詳細資訊頁面上，選擇「動作」。
    b. 如果您要新增訓練資料集，請選擇 [建立訓練資料集]。

c. 如果要新增測試資料集，請選擇 [建立測試資料集]。

d. 在 [建立資料集] 頁面上，輸入影像來源資訊。如需詳細資訊，請參閱the section called “建立包含影像的資料集”。

e. 選擇 [建立資料集] 以建立資料集。

8. 將影像新增至資料集。如需詳細資訊，請參閱添加更多圖像（控制台）。

9. 在資料集中新增標籤。如需詳細資訊，請參閱新增標籤 (主控台)。

10. 為您的圖像添加標籤。如果要新增影像層級標籤，請參閱the section called “將影像層級標籤指派給影像”。如果您要新增邊界方框，請參閱使用週框方塊標記物件。如需詳細資訊，請參閱規劃資料集。

將資料集新增至專案 (SDK)

您可以使用下列方式，將訓練或測試資料集新增至現有專案：

- 使用資訊清單檔案建立資料集。如需詳細資訊，請參閱使用 SageMaker Ground Truth 資訊清單檔案 (SDK) 建立資料集。
- 建立空白資料集並在之後填入資料集。下列範例顯示如何建立空的資料集。若要在建立空白資料集之後新增項目，請參閱將更多影像新增至資料集。

若要將資料集新增至專案 (SDK)

1. 若您尚未這樣做，請安裝AWS CLI並設定和AWS SDK。如需詳細資訊，請參閱步驟 4：設定 AWS CLI 和 AWS SDKs。

2. 使用下列範例將 JSON 行新增至資料集。

   CLI

   以`project_arn`您要新增資料集的專案。取代`TRAIN`為`dataset_type`以建立訓練資料集，或`TEST`建立測試資料集。

   ```
   aws rekognition create-dataset --project-arn project_arn \
     --dataset-type dataset_type \
     --profile custom-labels-access
   ```

   Python

   使用下列程式碼建立資料集。提供下列命令列選項：

- project_arn— 您要向其新增測試資料集的專案的 ARN。
- type— 您要建立 (訓練或測試) 的資料集類型

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

import argparse
import logging
import time
import boto3

from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)

def create_empty_dataset(rek_client, project_arn, dataset_type):
    """
    Creates an empty Amazon Rekognition Custom Labels dataset.
    :param rek_client: The Amazon Rekognition Custom Labels Boto3 client.
    :param project_arn: The ARN of the project in which you want to create a
 dataset.
    :param dataset_type: The type of the dataset that you want to create (train
 or test).
    """

    try:
        #Create the dataset.
        logger.info("Creating empty %s dataset for project %s",
            dataset_type, project_arn)

        dataset_type=dataset_type.upper()

        response = rek_client.create_dataset(
            ProjectArn=project_arn, DatasetType=dataset_type
        )

        dataset_arn=response['DatasetArn']

        logger.info("dataset ARN: %s", dataset_arn)

        finished=False
```

```
        while finished is False:

            dataset=rek_client.describe_dataset(DatasetArn=dataset_arn)

            status=dataset['DatasetDescription']['Status']

            if status == "CREATE_IN_PROGRESS":

                logger.info(("Creating dataset: %s ", dataset_arn))
                time.sleep(5)
                continue

            if status == "CREATE_COMPLETE":
                logger.info("Dataset created: %s", dataset_arn)
                finished=True
                continue

            if status == "CREATE_FAILED":
                error_message = f"Dataset creation failed: {status} :
 {dataset_arn}"
                logger.exception(error_message)
                raise Exception(error_message)

            error_message = f"Failed. Unexpected state for dataset creation:
 {status} : {dataset_arn}"
            logger.exception(error_message)
            raise Exception(error_message)

        return dataset_arn

    except ClientError as err:
        logger.exception("Couldn't create dataset: %s", err.response['Error']
['Message'])
        raise

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

    parser.add_argument(
        "project_arn", help="The ARN of the project in which you want to create
 the empty dataset."
```

```
    )

    parser.add_argument(
        "dataset_type", help="The type of the empty dataset that you want to
 create (train or test)."
    )


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()

        print(f"Creating empty {args.dataset_type} dataset for project
 {args.project_arn}")

        # Create the empty dataset.
        session = boto3.Session(profile_name='custom-labels-access')
        rekognition_client = session.client("rekognition")

        dataset_arn=create_empty_dataset(rekognition_client,
            args.project_arn,
            args.dataset_type.lower())

        print(f"Finished creating empty dataset: {dataset_arn}")


    except ClientError as err:
        logger.exception("Problem creating empty dataset: %s", err)
        print(f"Problem creating empty dataset: {err}")
    except Exception as err:
        logger.exception("Problem creating empty dataset: %s", err)
        print(f"Problem creating empty dataset: {err}")


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

Java V2

使用下列程式碼建立資料集。提供下列命令列選項：

- project_arn— 您要向其新增測試資料集的專案的 ARN。
- type— 您要建立 (訓練或測試) 的資料集類型

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

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.CreateDatasetRequest;
import software.amazon.awssdk.services.rekognition.model.CreateDatasetResponse;
import software.amazon.awssdk.services.rekognition.model.DatasetDescription;
import software.amazon.awssdk.services.rekognition.model.DatasetStatus;
import software.amazon.awssdk.services.rekognition.model.DatasetType;
import software.amazon.awssdk.services.rekognition.model.DescribeDatasetRequest;
import
 software.amazon.awssdk.services.rekognition.model.DescribeDatasetResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;

import java.net.URI;
import java.util.logging.Level;
import java.util.logging.Logger;

public class CreateEmptyDataset {

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

    public static String createMyEmptyDataset(RekognitionClient rekClient,
 String projectArn, String datasetType)
            throws Exception, RekognitionException {

        try {
```

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

            DatasetType requestDatasetType = null;

            switch (datasetType) {
            case "train":
                requestDatasetType = DatasetType.TRAIN;
                break;
            case "test":
                requestDatasetType = DatasetType.TEST;
                break;
            default:
                logger.log(Level.SEVERE, "Unrecognized dataset type: {0}",
datasetType);
                throw new Exception("Unrecognized dataset type: " +
datasetType);

            }

            CreateDatasetRequest createDatasetRequest =
CreateDatasetRequest.builder().projectArn(projectArn)
                    .datasetType(requestDatasetType).build();

            CreateDatasetResponse response =
rekClient.createDataset(createDatasetRequest);

            boolean created = false;

            //Wait until updates finishes

            do {

                DescribeDatasetRequest describeDatasetRequest =
DescribeDatasetRequest.builder()
                        .datasetArn(response.datasetArn()).build();
                DescribeDatasetResponse describeDatasetResponse =
rekClient.describeDataset(describeDatasetRequest);

                DatasetDescription datasetDescription =
describeDatasetResponse.datasetDescription();
```

```
                DatasetStatus status = datasetDescription.status();

                logger.log(Level.INFO, "Creating dataset ARN: {0} ",
response.datasetArn());

                switch (status) {

                case CREATE_COMPLETE:
                    logger.log(Level.INFO, "Dataset created");
                    created = true;
                    break;

                case CREATE_IN_PROGRESS:
                    Thread.sleep(5000);
                    break;

                case CREATE_FAILED:
                    String error = "Dataset creation failed: " +
datasetDescription.statusAsString() + " "
                            + datasetDescription.statusMessage() + " " +
response.datasetArn();
                    logger.log(Level.SEVERE, error);
                    throw new Exception(error);

                default:
                    String unexpectedError = "Unexpected creation state: " +
datasetDescription.statusAsString() + " "
                            + datasetDescription.statusMessage() + " " +
response.datasetArn();
                    logger.log(Level.SEVERE, unexpectedError);
                    throw new Exception(unexpectedError);
                }

            } while (created == false);

            return response.datasetArn();

        } catch (RekognitionException e) {
            logger.log(Level.SEVERE, "Could not create dataset: {0}",
e.getMessage());
            throw e;
        }

    }
```

```
    public static void main(String args[]) {

        String datasetType = null;
        String datasetArn = null;
        String projectArn = null;


        final String USAGE = "\n" + "Usage: " + "<project_arn> <dataset_type>\n
\n" + "Where:\n"
                + "   project_arn - the ARN of the project that you want to add
 copy the datast to.\n\n"
                + "   dataset_type - the type of the empty dataset that you want
 to create (train or test).\n\n";


        if (args.length != 2) {
            System.out.println(USAGE);
            System.exit(1);
        }

        projectArn = args[0];
        datasetType = args[1];

        try {

            // Get the Rekognition client
            RekognitionClient rekClient = RekognitionClient.builder()
                .credentialsProvider(ProfileCredentialsProvider.create("custom-
labels-access"))
                .region(Region.US_WEST_2)
                .build();

            // Create the dataset
            datasetArn = createMyEmptyDataset(rekClient, projectArn,
 datasetType);

            System.out.println(String.format("Created dataset: %s",
 datasetArn));

            rekClient.close();

        } catch (RekognitionException rekError) {
```

```
            logger.log(Level.SEVERE, "Rekognition client error: {0}",
 rekError.getMessage());
            System.exit(1);
        } catch (Exception rekError) {
            logger.log(Level.SEVERE, "Error: {0}", rekError.getMessage());
            System.exit(1);
        }

    }

}
```

3. 將影像新增至資料集。如需詳細資訊，請參閱新增更多影像 (SDK)。

將更多影像新增至資料集

您可以使用 Amazon Rekognition 自訂標籤主控台或呼叫UpdateDatasetEntries API，將更多影像新增至資料集。

主題

- 添加更多圖像（控制台）
- 新增更多影像 (SDK)

添加更多圖像（控制台）

當您使用 Amazon Rekognition 自訂標籤主控台時，您可以從本機電腦上傳影像。映像檔會新增至 Amazon S3 儲存貯體位置 (主控台或外部)，用於建立資料集的映像存放在該位置。

若要將更多影像新增至資料集 (主控台)

1. 開啟亞馬遜重新認知主控台，網址為 https://console.aws.amazon.com/rekognition/。
2. 在左窗格中選擇使用自訂標籤。顯示 Amazon Rekognition 自訂標籤登陸頁面。
3. 在左側導覽窗格中選擇 Projects (專案)。將顯示「專案」檢視。
4. 選擇您要使用的專案。
5. 在左側導覽窗格的專案名稱下，選擇 Databet (資料集)。
6. 選擇 Actions (動作)，然後選取要新增影像的資料集。
7. 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。

8. 選擇 [上傳圖片]。
9. 選擇 Save changes (儲存變更)。
10. 標記影像。如需詳細資訊，請參閱標記檔案。

新增更多影像 (SDK)

UpdateDatasetEntries更新或將 JSON 行添加到清單文件中。您將 JSON 行作為GroundTruth字段中的字節 64 編碼數據對象傳遞。如果您使用AWS SDK 來呼叫UpdateDatasetEntries，SDK 會為您編碼資料。每個 JSON 行都包含單一影像的資訊，例如指派的標籤或邊界方框資訊。例如：

```
{"source-ref":"s3://bucket/image","BB":{"annotations":
[{"left":1849,"top":1039,"width":422,"height":283,"class_id":0},
{"left":1849,"top":1340,"width":443,"height":415,"class_id":1},
{"left":2637,"top":1380,"width":676,"height":338,"class_id":2},
{"left":2634,"top":1051,"width":673,"height":338,"class_id":3}],"image_size":
[{"width":4000,"height":2667,"depth":3}]},"BB-metadata":{"job-name":"labeling-job/
BB","class-map":
{"0":"comparator","1":"pot_resistor","2":"ir_phototransistor","3":"ir_led"},"human-
annotated":"yes","objects":[{"confidence":1},{"confidence":1},{"confidence":1},
{"confidence":1}],"creation-date":"2021-06-22T10:11:18.006Z","type":"groundtruth/
object-detection"}}
```

如需詳細資訊，請參閱建立清單檔案。

使用source-ref欄位作為關鍵字，以識別您要更新的影像。如果資料集不包含相符的source-ref欄位值，則會將 JSON 行新增為新影像。

若要將更多影像新增至資料集 (SDK)

1. 若您尚未這樣做，請安裝AWS CLI並設定和AWS SDK。如需詳細資訊，請參閱步驟 4：設定 AWS CLI 和 AWS SDKs。
2. 使用下列範例將 JSON 行新增至資料集。

   CLI

   以您要使GroundTruth用的 JSON 行取代的值。您需要轉義 JSON 行中的任何特殊字符。

   ```
   aws rekognition update-dataset-entries\
     --dataset-arn dataset_arn \
   ```

```
  --changes '{"GroundTruth" : "{\"source-ref\":\"s3://your_bucket/your_image
\",\"BB\":{\"annotations\":[{\"left\":1776,\"top\":1017,\"width\":458,\"height
\":317,\"class_id\":0},{\"left\":1797,\"top\":1334,\"width\":418,\"height
\":415,\"class_id\":1},{\"left\":2597,\"top\":1361,\"width\":655,\"height
\":329,\"class_id\":2},{\"left\":2581,\"top\":1020,\"width\":689,\"height
\":338,\"class_id\":3}],\"image_size\":[{\"width\":4000,\"height\":2667,
\"depth\":3}]},\"BB-metadata\":{\"job-name\":\"labeling-job/BB\",\"class-map
\":{\"0\":\"comparator\",\"1\":\"pot_resistor\",\"2\":\"ir_phototransistor\",
\"3\":\"ir_led\"},\"human-annotated\":\"yes\",\"objects\":[{\"confidence\":1},
{\"confidence\":1},{\"confidence\":1},{\"confidence\":1}],\"creation-date\":
\"2021-06-22T10:10:48.492Z\",\"type\":\"groundtruth/object-detection\"}}" }' \
  --cli-binary-format raw-in-base64-out \
  --profile custom-labels-access
```

Python

使用下列程式碼。提供以下命令行參數：

- 資料集-您要更新之資料集的 ARN。
- 更新文件-包含 JSON 行更新的文件。

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

"""
Purpose
Shows how to add entries to an Amazon Rekognition Custom Labels dataset.
"""

import argparse
import logging
import time
import json
import boto3

from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)

def update_dataset_entries(rek_client, dataset_arn, updates_file):
    """
    Adds dataset entries to an Amazon Rekognition Custom Labels dataset.
```

```
    :param rek_client: The Amazon Rekognition Custom Labels Boto3 client.
    :param dataset_arn: The ARN of the dataset that yuo want to update.
    :param updates_file: The manifest file of JSON Lines that contains the
updates.
    """

    try:
        status=""
        status_message=""

        # Update dataset entries.
        logger.info("Updating dataset %s", dataset_arn)


        with open(updates_file) as f:
            manifest_file = f.read()


        changes=json.loads('{ "GroundTruth" : ' +
            json.dumps(manifest_file) +
            '}')

        rek_client.update_dataset_entries(
            Changes=changes, DatasetArn=dataset_arn
        )

        finished=False
        while finished is False:

            dataset=rek_client.describe_dataset(DatasetArn=dataset_arn)

            status=dataset['DatasetDescription']['Status']
            status_message=dataset['DatasetDescription']['StatusMessage']

            if status == "UPDATE_IN_PROGRESS":

                logger.info("Updating dataset: %s ", dataset_arn)
                time.sleep(5)
                continue

            if status == "UPDATE_COMPLETE":
                logger.info("Dataset updated: %s : %s : %s",
                    status, status_message, dataset_arn)
                finished=True
```

```
                continue

            if status == "UPDATE_FAILED":
                error_message = f"Dataset update failed: {status} :
 {status_message} : {dataset_arn}"
                logger.exception(error_message)
                raise Exception (error_message)

            error_message = f"Failed. Unexpected state for dataset update:
 {status} : {status_message} : {dataset_arn}"
            logger.exception(error_message)
            raise Exception(error_message)

        logger.info("Added entries to dataset")

        return status, status_message


    except ClientError as err:
        logger.exception("Couldn't update dataset: %s", err.response['Error']
['Message'])
        raise

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

    parser.add_argument(
        "dataset_arn", help="The ARN of the dataset that you want to update."
    )

    parser.add_argument(
        "updates_file", help="The manifest file of JSON Lines that contains the
 updates."
    )

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()

        print(f"Updating dataset {args.dataset_arn} with entries from
 {args.updates_file}.")

        # Update the dataset.
        session = boto3.Session(profile_name='custom-labels-access')
        rekognition_client = session.client("rekognition")

        status, status_message=update_dataset_entries(rekognition_client,
            args.dataset_arn,
            args.updates_file)

        print(f"Finished updates dataset: {status} : {status_message}")


    except ClientError as err:
        logger.exception("Problem updating dataset: %s", err)
        print(f"Problem updating dataset: {err}")

    except Exception as err:
        logger.exception("Problem updating dataset: %s", err)
        print(f"Problem updating dataset: {err}")


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

Java V2

- 資料集-您要更新之資料集的 ARN。
- 更新文件-包含 JSON 行更新的文件。

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

```
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.DatasetChanges;
import software.amazon.awssdk.services.rekognition.model.DatasetDescription;
import software.amazon.awssdk.services.rekognition.model.DatasetStatus;
import software.amazon.awssdk.services.rekognition.model.DescribeDatasetRequest;
import
 software.amazon.awssdk.services.rekognition.model.DescribeDatasetResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import
 software.amazon.awssdk.services.rekognition.model.UpdateDatasetEntriesRequest;
import
 software.amazon.awssdk.services.rekognition.model.UpdateDatasetEntriesResponse;

import java.io.FileInputStream;
import java.io.InputStream;
import java.util.logging.Level;
import java.util.logging.Logger;

public class UpdateDatasetEntries {

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

    public static String updateMyDataset(RekognitionClient rekClient, String
 datasetArn,
            String updateFile
            ) throws Exception, RekognitionException {

        try {

            logger.log(Level.INFO, "Updating dataset {0}",
                    new Object[] { datasetArn});


            InputStream sourceStream = new FileInputStream(updateFile);
            SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);

            DatasetChanges datasetChanges = DatasetChanges.builder()
                    .groundTruth(sourceBytes).build();
```

```
            UpdateDatasetEntriesRequest updateDatasetEntriesRequest =
UpdateDatasetEntriesRequest.builder()
                    .changes(datasetChanges)
                    .datasetArn(datasetArn)
                    .build();

            UpdateDatasetEntriesResponse response =
rekClient.updateDatasetEntries(updateDatasetEntriesRequest);

            boolean updated = false;

            //Wait until update completes

            do {

                DescribeDatasetRequest describeDatasetRequest =
DescribeDatasetRequest.builder()
                        .datasetArn(datasetArn).build();
                DescribeDatasetResponse describeDatasetResponse =
rekClient.describeDataset(describeDatasetRequest);

                DatasetDescription datasetDescription =
describeDatasetResponse.datasetDescription();

                DatasetStatus status = datasetDescription.status();

                logger.log(Level.INFO, " dataset ARN: {0} ", datasetArn);

                switch (status) {

                case UPDATE_COMPLETE:
                    logger.log(Level.INFO, "Dataset updated");
                    updated = true;
                    break;

                case UPDATE_IN_PROGRESS:
                    Thread.sleep(5000);
                    break;

                case UPDATE_FAILED:
                    String error = "Dataset update failed: " +
datasetDescription.statusAsString() + " "
                            + datasetDescription.statusMessage() + " " +
datasetArn;
```

```
                    logger.log(Level.SEVERE, error);
                    throw new Exception(error);

                default:
                    String unexpectedError = "Unexpected update state: " +
 datasetDescription.statusAsString() + " "
                            + datasetDescription.statusMessage() + " " +
 datasetArn;
                    logger.log(Level.SEVERE, unexpectedError);
                    throw new Exception(unexpectedError);
                }

            } while (updated == false);

            return datasetArn;

        } catch (RekognitionException e) {
            logger.log(Level.SEVERE, "Could not update dataset: {0}",
 e.getMessage());
            throw e;
        }

    }

    public static void main(String args[]) {

        String updatesFile = null;
        String datasetArn = null;


        final String USAGE = "\n" + "Usage: " + "<project_arn> <dataset_arn>
 <updates_file>\n\n" + "Where:\n"
                + "   dataset_arn - the ARN of the dataset that you want to
 update.\n\n"
                + "   update_file - The file that includes in JSON Line updates.
\n\n";

        if (args.length != 2) {
            System.out.println(USAGE);
            System.exit(1);
        }

        datasetArn = args[0];
        updatesFile = args[1];
```

```
        try {

            // Get the Rekognition client.
            RekognitionClient rekClient = RekognitionClient.builder()
                .credentialsProvider(ProfileCredentialsProvider.create("custom-
labels-access"))
                .region(Region.US_WEST_2)
                .build();

             // Update the dataset
            datasetArn = updateMyDataset(rekClient, datasetArn, updatesFile);

            System.out.println(String.format("Dataset updated: %s",
 datasetArn));

            rekClient.close();

        } catch (RekognitionException rekError) {
            logger.log(Level.SEVERE, "Rekognition client error: {0}",
 rekError.getMessage());
            System.exit(1);
        } catch (Exception rekError) {
            logger.log(Level.SEVERE, "Error: {0}", rekError.getMessage());
            System.exit(1);
        }

    }

}
```

使用現有資料集 (SDK) 建立資料集

下列程序顯示如何使用CreateDataset作業從現有資料集的建立資料集。

1. 若您尚未這樣做，請安裝AWS CLI並設定和AWS SDK。如需詳細資訊，請參閱步驟 4：設定 AWS CLI 和 AWS SDKs。
2. 使用下列範例程式碼，透過複製另一個資料集來建立資料集。

AWS CLI

使用下列程式碼建立資料集。取代下列項目：

- project_arn— 您要新增資料集的專案的 ARN。
- dataset_type— 您要在專案中建立的資料集類型 (TRAIN或TEST)。
- dataset_arn— 使用您要複製之資料集的 ARN。

```
aws rekognition create-dataset --project-arn project_arn \
  --dataset-type dataset_type \
  --dataset-source '{ "DatasetArn" : "dataset_arn" }' \
  --profile custom-labels-access
```

Python

下列範例會使用現有資料集建立資料集，並顯示其 ARN。

若要執行程式，請提供下列命令列引數：

- project_arn— 您要使用的專案的 ARN。
- dataset_type— 您要建立的專案資料集類型 (train或test)。
- dataset_arn— 您要建立資料集的資料集的 ARN。

```
# Copyright 2023 Amazon.com, Inc. or its affiliates. All Rights Reserved.
# PDX-License-Identifier: MIT-0 (For details, see https://github.com/
awsdocs/amazon-rekognition-custom-labels-developer-guide/blob/master/LICENSE-
SAMPLECODE.)

import argparse
import logging
import time
import json
import boto3

from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)
```

```
def create_dataset_from_existing_dataset(rek_client, project_arn, dataset_type,
 dataset_arn):
    """
    Creates an Amazon Rekognition Custom Labels dataset using an existing
 dataset.
    :param rek_client: The Amazon Rekognition Custom Labels Boto3 client.
    :param project_arn: The ARN of the project in which you want to create a
 dataset.
    :param dataset_type: The type of the dataset that you want to create (train
 or test).
    :param dataset_arn: The ARN of the existing dataset that you want to use.
    """

    try:
        # Create the dataset

        dataset_type=dataset_type.upper()

        logger.info(
            "Creating %s dataset for project %s from dataset %s.",
                dataset_type,project_arn, dataset_arn)

        dataset_source = json.loads(
            '{ "DatasetArn": "' + dataset_arn + '"}'
        )

        response = rek_client.create_dataset(
            ProjectArn=project_arn, DatasetType=dataset_type,
 DatasetSource=dataset_source
        )

        dataset_arn = response['DatasetArn']

        logger.info("New dataset ARN: %s", dataset_arn)

        finished = False
        while finished is False:

            dataset = rek_client.describe_dataset(DatasetArn=dataset_arn)

            status = dataset['DatasetDescription']['Status']

            if status == "CREATE_IN_PROGRESS":
```

```
                logger.info(("Creating dataset: %s ", dataset_arn))
                time.sleep(5)
                continue

            if status == "CREATE_COMPLETE":
                logger.info("Dataset created: %s", dataset_arn)
                finished = True
                continue

            if status == "CREATE_FAILED":
                error_message = f"Dataset creation failed: {status} :
 {dataset_arn}"
                logger.exception(error_message)
                raise Exception(error_message)

            error_message = f"Failed. Unexpected state for dataset creation:
 {status} : {dataset_arn}"
            logger.exception(error_message)
            raise Exception(error_message)

        return dataset_arn

    except ClientError as err:
        logger.exception(
            "Couldn't create dataset: %s",err.response['Error']['Message'] )
        raise


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

    parser.add_argument(
        "project_arn", help="The ARN of the project in which you want to create
 the dataset."
    )

    parser.add_argument(
        "dataset_type", help="The type of the dataset that you want to create
 (train or test)."
    )
```

```
    parser.add_argument(
        "dataset_arn", help="The ARN of the dataset that you want to copy from."
    )


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()

        print(
            f"Creating {args.dataset_type} dataset for project
 {args.project_arn}")

        # Create the dataset.
        session = boto3.Session(profile_name='custom-labels-access')
        rekognition_client = session.client("rekognition")

        dataset_arn = create_dataset_from_existing_dataset(rekognition_client,
                                    args.project_arn,
                                    args.dataset_type,
                                    args.dataset_arn)

        print(f"Finished creating dataset: {dataset_arn}")

    except ClientError as err:
        logger.exception("Problem creating dataset: %s", err)
        print(f"Problem creating dataset: {err}")
    except Exception as err:
        logger.exception("Problem creating dataset: %s", err)
        print(f"Problem creating dataset: {err}")


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

Java V2

下列範例會使用現有資料集建立資料集，並顯示其 ARN。

若要執行程式，請提供下列命令列引數：

- project_arn— 您要使用的專案的 ARN。
- dataset_type— 您要建立的專案資料集類型 (train或test)。
- dataset_arn— 您要建立資料集的資料集的 ARN。

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

package com.example.rekognition;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.CreateDatasetRequest;
import software.amazon.awssdk.services.rekognition.model.CreateDatasetResponse;
import software.amazon.awssdk.services.rekognition.model.DatasetDescription;
import software.amazon.awssdk.services.rekognition.model.DatasetSource;
import software.amazon.awssdk.services.rekognition.model.DatasetStatus;
import software.amazon.awssdk.services.rekognition.model.DatasetType;
import software.amazon.awssdk.services.rekognition.model.DescribeDatasetRequest;
import
 software.amazon.awssdk.services.rekognition.model.DescribeDatasetResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;

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

public class CreateDatasetExisting {

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

    public static String createMyDataset(RekognitionClient rekClient, String
 projectArn, String datasetType,
```

```
            String existingDatasetArn) throws Exception, RekognitionException {

       try {

           logger.log(Level.INFO, "Creating {0} dataset for project : {1} from
dataset {2} ",
                   new Object[] { datasetType.toString(), projectArn,
existingDatasetArn });

           DatasetType requestDatasetType = null;

           switch (datasetType) {
           case "train":
               requestDatasetType = DatasetType.TRAIN;
               break;
           case "test":
               requestDatasetType = DatasetType.TEST;
               break;
           default:
               logger.log(Level.SEVERE, "Unrecognized dataset type: {0}",
datasetType);
               throw new Exception("Unrecognized dataset type: " +
datasetType);

           }

           DatasetSource datasetSource =
DatasetSource.builder().datasetArn(existingDatasetArn).build();

           CreateDatasetRequest createDatasetRequest =
CreateDatasetRequest.builder().projectArn(projectArn)

.datasetType(requestDatasetType).datasetSource(datasetSource).build();

           CreateDatasetResponse response =
rekClient.createDataset(createDatasetRequest);

           boolean created = false;

           //Wait until create finishes

           do {
```

```
                DescribeDatasetRequest describeDatasetRequest =
DescribeDatasetRequest.builder()
                        .datasetArn(response.datasetArn()).build();
                DescribeDatasetResponse describeDatasetResponse =
rekClient.describeDataset(describeDatasetRequest);

                DatasetDescription datasetDescription =
describeDatasetResponse.datasetDescription();

                DatasetStatus status = datasetDescription.status();

                logger.log(Level.INFO, "Creating dataset ARN: {0} ",
response.datasetArn());

                switch (status) {

                case CREATE_COMPLETE:
                    logger.log(Level.INFO, "Dataset created");
                    created = true;
                    break;

                case CREATE_IN_PROGRESS:
                    Thread.sleep(5000);
                    break;

                case CREATE_FAILED:
                    String error = "Dataset creation failed: " +
datasetDescription.statusAsString() + " "
                            + datasetDescription.statusMessage() + " " +
response.datasetArn();
                    logger.log(Level.SEVERE, error);
                    throw new Exception(error);

                default:
                    String unexpectedError = "Unexpected creation state: " +
datasetDescription.statusAsString() + " "
                            + datasetDescription.statusMessage() + " " +
response.datasetArn();
                    logger.log(Level.SEVERE, unexpectedError);
                    throw new Exception(unexpectedError);
                }

            } while (created == false);
```

```
            return response.datasetArn();

        } catch (RekognitionException e) {
            logger.log(Level.SEVERE, "Could not create dataset: {0}",
 e.getMessage());
            throw e;
        }

    }

    public static void main(String[] args) {

        String datasetType = null;
        String datasetArn = null;
        String projectArn = null;
        String datasetSourceArn = null;

        final String USAGE = "\n" + "Usage: " + "<project_arn> <dataset_type>
 <dataset_arn>\n\n" + "Where:\n"
                + "   project_arn - the ARN of the project that you want to add
 copy the datast to.\n\n"
                + "   dataset_type - the type of the dataset that you want to
 create (train or test).\n\n"
                + "   dataset_arn - the ARN of the dataset that you want to copy
 from.\n\n";

        if (args.length != 3) {
            System.out.println(USAGE);
            System.exit(1);
        }

        projectArn = args[0];
        datasetType = args[1];
        datasetSourceArn = args[2];

        try {

            // Get the Rekognition client
            RekognitionClient rekClient = RekognitionClient.builder()
                .credentialsProvider(ProfileCredentialsProvider.create("custom-
labels-access"))
                .region(Region.US_WEST_2)
                .build();
```

```
            // Create the dataset
            datasetArn = createMyDataset(rekClient, projectArn, datasetType,
 datasetSourceArn);

            System.out.println(String.format("Created dataset: %s",
 datasetArn));

            rekClient.close();

        } catch (RekognitionException rekError) {
            logger.log(Level.SEVERE, "Rekognition client error: {0}",
 rekError.getMessage());
            System.exit(1);
        } catch (Exception rekError) {
            logger.log(Level.SEVERE, "Error: {0}", rekError.getMessage());
            System.exit(1);
        }

    }

}
```

描述資料集 (SDK)

您可以使用`DescribeDataset` API 來取得資料集的相關資訊。

若要描述資料集 (SDK)

1. 若果您尚未這樣做，請先完成安裝AWS CLI並設定和AWS SDK。如需詳細資訊，請參閱步驟 4：設定 AWS CLI 和 AWS SDKs。
2. 使用下列範例程式碼來描述資料集。

   AWS CLI

   `dataset-arn`將的值變更為您要描述之資料集的 ARN。

```
aws rekognition describe-dataset --dataset-arn dataset_arn \
  --profile custom-labels-access
```

Python

使用下列程式碼。提供以下命令行參數：

- 資料集-您要描述之資料集的 ARN。

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

"""
Purpose
Shows how to describe an Amazon Rekognition Custom Labels dataset.
"""

import argparse
import logging
import boto3

from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)


def describe_dataset(rek_client, dataset_arn):
    """
    Describes an Amazon Rekognition Custom Labels dataset.
    :param rek_client: The Amazon Rekognition Custom Labels Boto3 client.
    :param dataset_arn: The ARN of the dataset that you want to describe.

    """

    try:
        # Describe the dataset
        logger.info("Describing dataset %s", dataset_arn)

        dataset = rek_client.describe_dataset(DatasetArn=dataset_arn)

        description = dataset['DatasetDescription']

        print(f"Created: {str(description['CreationTimestamp'])}")
        print(f"Updated: {str(description['LastUpdatedTimestamp'])}")
        print(f"Status: {description['Status']}")
```

```
        print(f"Status message: {description['StatusMessage']}")
        print(f"Status code: {description['StatusMessageCode']}")
        print("Stats:")
        print(
            f"\tLabeled entries: {description['DatasetStats']
['LabeledEntries']}")
        print(
            f"\tTotal entries: {description['DatasetStats']['TotalEntries']}")
        print(f"\tTotal labels: {description['DatasetStats']['TotalLabels']}")

    except ClientError as err:
        logger.exception("Couldn't describe dataset: %s",
                         err.response['Error']['Message'])
        raise


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

    parser.add_argument(
        "dataset_arn", help="The ARN of the dataset that you want to describe."
    )


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()

        print(f"Describing dataset {args.dataset_arn}")

        # Describe the dataset.
        session = boto3.Session(profile_name='custom-labels-access')
        rekognition_client = session.client("rekognition")
```

```
        describe_dataset(rekognition_client, args.dataset_arn)

        print(f"Finished describing dataset: {args.dataset_arn}")

    except ClientError as err:
        error_message=f"Problem describing dataset: {err}"
        logger.exception(error_message)
        print(error_message)
    except Exception as err:
        error_message = f"Problem describing dataset: {err}"
        logger.exception(error_message)
        print(error_message)


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

Java V2

- 資料集-您要描述之資料集的 ARN。

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

package com.example.rekognition;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.DatasetDescription;
import software.amazon.awssdk.services.rekognition.model.DatasetStats;
import software.amazon.awssdk.services.rekognition.model.DescribeDatasetRequest;
import
 software.amazon.awssdk.services.rekognition.model.DescribeDatasetResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;

import java.util.logging.Level;
import java.util.logging.Logger;
```

```
public class DescribeDataset {

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

   public static void describeMyDataset(RekognitionClient rekClient, String
datasetArn) {

       try {

           DescribeDatasetRequest describeDatasetRequest =
DescribeDatasetRequest.builder().datasetArn(datasetArn)
                   .build();
           DescribeDatasetResponse describeDatasetResponse =
rekClient.describeDataset(describeDatasetRequest);

           DatasetDescription datasetDescription =
describeDatasetResponse.datasetDescription();
           DatasetStats datasetStats = datasetDescription.datasetStats();

           System.out.println("ARN: " + datasetArn);
           System.out.println("Created: " +
datasetDescription.creationTimestamp().toString());
           System.out.println("Updated: " +
datasetDescription.lastUpdatedTimestamp().toString());
           System.out.println("Status: " +
datasetDescription.statusAsString());
           System.out.println("Message: " +
datasetDescription.statusMessage());
           System.out.println("Total Labels: " +
datasetStats.totalLabels().toString());
           System.out.println("Total entries: " +
datasetStats.totalEntries().toString());
           System.out.println("Entries with labels: " +
datasetStats.labeledEntries().toString());
           System.out.println("Entries with at least 1 error: " +
datasetStats.errorEntries().toString());

       } catch (RekognitionException rekError) {
           logger.log(Level.SEVERE, "Rekognition client error: {0}",
rekError.getMessage());
           throw rekError;
       }
```

```
    }

    public static void main(String[] args) {

        final String USAGE = "\n" + "Usage: " + "<dataset_arn>\n\n" + "Where:\n"
                + "   dataset_arn - The ARN of the dataset that you want to
 describe.\n\n";

        if (args.length != 1) {
            System.out.println(USAGE);
            System.exit(1);
        }

        String datasetArn = args[0];

        try {

            // Get the Rekognition client.
            RekognitionClient rekClient = RekognitionClient.builder()
            .credentialsProvider(ProfileCredentialsProvider.create("custom-
labels-access"))
            .region(Region.US_WEST_2)
            .build();

             // Describe the dataset.
            describeMyDataset(rekClient, datasetArn);

            rekClient.close();

        } catch (RekognitionException rekError) {
            logger.log(Level.SEVERE, "Rekognition client error: {0}",
 rekError.getMessage());
            System.exit(1);
        }

    }

}
```

列出資料集項目 (SDK)

您可以使用ListDatasetEntries API 列出資料集中每個影像的 JSON 行。如需詳細資訊，請參閱建立清單檔案。

若要列出資料集項目 (SDK)

1. 若果您尚未這樣做，請先完成安裝AWS CLI並設定和AWS SDK。如需詳細資訊，請參閱步驟 4：設定 AWS CLI 和 AWS SDKs。
2. 使用下列範例程式碼會列出資料集中的項目

   AWS CLI

   dataset-arn將的值變更為您要列出的資料集的 ARN。

   ```
   aws rekognition list-dataset-entries --dataset-arn dataset_arn \
     --profile custom-labels-access
   ```

   若只要列出含有錯誤的 JSON 行，請指定has-errors。

   ```
   aws rekognition list-dataset-entries --dataset-arn dataset_arn \
     --has-errors \
     --profile custom-labels-access
   ```

   Python

   使用下列程式碼。提供以下命令行參數：

   - 資料集-您要列出之資料集的 ARN。
   - 僅顯示錯誤 — 指定您是true否只想查看錯誤。 false否則。

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

   """
   Purpose
   Shows how to list the entries in an Amazon Rekognition Custom Labels dataset.
   """

   import argparse
   ```

```
import logging
import boto3

from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)


def list_dataset_entries(rek_client, dataset_arn, show_errors):
    """
    Lists the entries in an Amazon Rekognition Custom Labels dataset.
    :param rek_client: The Amazon Rekognition Custom Labels Boto3 client.
    :param dataset_arn: The ARN of the dataet that you want to use.
    """

    try:
        # List the entries.
        logger.info("Listing dataset entries for the dataset %s.", dataset_arn)

        finished = False
        count = 0
        next_token = ""
        show_errors_only = False

        if show_errors.lower() == "true":
            show_errors_only = True

        while finished is False:

            response = rek_client.list_dataset_entries(
                DatasetArn=dataset_arn,
                HasErrors=show_errors_only,
                MaxResults=100,
                NextToken=next_token)

            count += len(response['DatasetEntries'])

            for entry in response['DatasetEntries']:
                print(entry)

            if 'NextToken' not in response:
                finished = True
                logger.info("No more entries. Total:%s", count)
            else:
```

```
                next_token = next_token = response['NextToken']
                logger.info("Getting more entries. Total so far :%s", count)

    except ClientError as err:
        logger.exception(
            "Couldn't list dataset: %s",
             err.response['Error']['Message'])
        raise


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

    parser.add_argument(
        "dataset_arn", help="The ARN of the dataset that you want to list."

    )

    parser.add_argument(
        "show_errors_only", help="true if you want to see errors only. false
 otherwise."
    )


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()

        print(f"Listing entries for  dataset {args.dataset_arn}")

        # List the dataset entries.
        session = boto3.Session(profile_name='custom-labels-access')
        rekognition_client = session.client("rekognition")
```

```
        list_dataset_entries(rekognition_client,
                             args.dataset_arn,
                             args.show_errors_only)

        print(f"Finished listing entries for dataset: {args.dataset_arn}")

    except ClientError as err:
        error_message = f"Problem listing dataset: {err}"
        logger.exception(error_message)
        print(error_message)
    except Exception as err:
        error_message = f"Problem listing dataset: {err}"
        logger.exception(error_message)
        print(error_message)


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

Java V2

使用下列程式碼。提供以下命令行參數：

- 資料集-您要列出之資料集的 ARN。
- 僅顯示錯誤 — 指定您是`true`否只想查看錯誤。 `false`否則。

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

package com.example.rekognition;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import
 software.amazon.awssdk.services.rekognition.model.ListDatasetEntriesRequest;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import
 software.amazon.awssdk.services.rekognition.paginators.ListDatasetEntriesIterable;
```

```
import java.net.URI;
import java.util.logging.Level;
import java.util.logging.Logger;

public class ListDatasetEntries {

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

   public static void listMyDatasetEntries(RekognitionClient rekClient, String
 datasetArn, boolean showErrorsOnly)
           throws Exception, RekognitionException {

       try {

           logger.log(Level.INFO, "Listing dataset {0}", new Object[]
 { datasetArn });

           ListDatasetEntriesRequest listDatasetEntriesRequest =
 ListDatasetEntriesRequest.builder()

 .hasErrors(showErrorsOnly).datasetArn(datasetArn).maxResults(1).build();

           ListDatasetEntriesIterable datasetEntriesList = rekClient
                   .listDatasetEntriesPaginator(listDatasetEntriesRequest);

           datasetEntriesList.stream().flatMap(r ->
 r.datasetEntries().stream())
                   .forEach(datasetEntry ->
 System.out.println(datasetEntry.toString()));

       } catch (RekognitionException e) {
           logger.log(Level.SEVERE, "Could not update dataset: {0}",
 e.getMessage());
           throw e;
       }

   }

   public static void main(String args[]) {

       boolean showErrorsOnly = false;
```

```
        String datasetArn = null;

        final String USAGE = "\n" + "Usage: " + "<project_arn> <dataset_arn>
 <updates_file>\n\n" + "Where:\n"
                + "   dataset_arn - the ARN of the dataset that you want to
 update.\n\n"
                + "   show_errors_only - true to show only errors. false
 otherwise.\n\n";

        if (args.length != 2) {
            System.out.println(USAGE);
            System.exit(1);
        }

        datasetArn = args[0];
        if (args[1].toLowerCase().equals("true")) {

            showErrorsOnly = true;
        }

        try {

            // Get the Rekognition client.
            RekognitionClient rekClient = RekognitionClient.builder()
            .credentialsProvider(ProfileCredentialsProvider.create("custom-
labels-access"))
            .region(Region.US_WEST_2)
            .build();

             // list the dataset entries.

            listMyDatasetEntries(rekClient, datasetArn, showErrorsOnly);

            System.out.println(String.format("Finished listing entries for :
 %s", datasetArn));

            rekClient.close();

        } catch (RekognitionException rekError) {
            logger.log(Level.SEVERE, "Rekognition client error: {0}",
 rekError.getMessage());
            System.exit(1);
        } catch (Exception rekError) {
            logger.log(Level.SEVERE, "Error: {0}", rekError.getMessage());
```

```
            System.exit(1);
        }

    }

}
```

發佈訓練資料集 (SDK)

Amazon Rekognition 自訂標籤需要訓練資料集和測試資料集來訓練您的模型。

如果您使用 API，則可以使用 DistributeDatasetEntriesAPI 將 20% 的訓練資料集散發到空白的測試資料集中。如果您只有一個資訊清單檔案可用，則散佈訓練資料集會很有用。使用單一資訊清單檔案建立訓練資料集。然後創建一個空的測試數據集並用`DistributeDatasetEntries`於填充測試數據集。

> **Note**
>
> 如果您使用 Amazon Rekognition 自訂標籤主控台並從單一資料集專案開始，Amazon Rekognition 自訂標籤會在訓練期間分割 (分配) 訓練資料集，以建立測試資料集。20% 的訓練資料集項目會移至測試資料集。

散發訓練資料集 (SDK)

1. 若果您尚未這樣做，請先完成安裝AWS CLI並設定和AWS SDK。如需詳細資訊，請參閱步驟 4：設定 AWS CLI 和 AWS SDKs。
2. 建立專案。如需詳細資訊，請參閱建立 Amazon Rekognition 自訂標籤專案 (SDK)。
3. 建立您的訓練資料集。如需資料集的相關資訊，請參閱建立訓練和測試資料集。
4. 建立空的測試資料集。
5. 使用下列範例程式碼，將 20% 的訓練資料集項目散佈至測試資料集。您可以透過呼叫取得專案資料集的 Amazon 資源名稱 (ARN) DescribeProjects。如需範例程式碼，請參閱描述一個項目（SDK）。

   AWS CLI

   使用您要使`training_dataset-arn`和`test_dataset_arn`用的資料集的 ARNS 來變更與的值。

```
aws rekognition distribute-dataset-entries --datasets ['{"Arn":
 "training_dataset_arn"}, {"Arn": "test_dataset_arn"}'] \
  --profile custom-labels-access
```

Python

使用下列程式碼。提供以下命令行參數：

- 訓練資料集 — 您從中發佈項目之訓練資料集的 ARN。
- test_dataset_arn — 您分發項目的測試資料集的 ARN。

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

import argparse
import logging
import time
import json
import boto3

from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)


def check_dataset_status(rek_client, dataset_arn):
    """
    Checks the current status of a dataset.
    :param rek_client: The Amazon Rekognition Custom Labels Boto3 client.
    :param dataset_arn: The dataset that you want to check.
    :return: The dataset status and status message.
    """
    finished = False
    status = ""
    status_message = ""

    while finished is False:

        dataset = rek_client.describe_dataset(DatasetArn=dataset_arn)

        status = dataset['DatasetDescription']['Status']
```

```
        status_message = dataset['DatasetDescription']['StatusMessage']

        if status == "UPDATE_IN_PROGRESS":

            logger.info("Distributing dataset: %s ", dataset_arn)
            time.sleep(5)
            continue

        if status == "UPDATE_COMPLETE":
            logger.info(
                "Dataset distribution complete: %s : %s : %s",
                    status, status_message, dataset_arn)
            finished = True
            continue

        if status == "UPDATE_FAILED":
            logger.exception(
                "Dataset distribution failed: %s : %s : %s",
                    status, status_message, dataset_arn)
            finished = True
            break

        logger.exception(
            "Failed. Unexpected state for dataset distribution: %s : %s : %s",
            status, status_message, dataset_arn)
        finished = True
        status_message = "An unexpected error occurred while distributing the
 dataset"
        break

    return status, status_message


def distribute_dataset_entries(rek_client, training_dataset_arn,
 test_dataset_arn):
    """
    Distributes 20% of the supplied training dataset into the supplied test
 dataset.
    :param rek_client: The Amazon Rekognition Custom Labels Boto3 client.
    :param training_dataset_arn: The ARN of the training dataset that you
 distribute entries from.
    :param test_dataset_arn: The ARN of the test dataset that you distribute
 entries to.
    """
```

```
    try:
        # List dataset labels.
        logger.info("Distributing training dataset entries (%s) into test
 dataset (%s).",
            training_dataset_arn,test_dataset_arn)


        datasets = json.loads(
            '[{"Arn" : "' + str(training_dataset_arn) + '"},{"Arn" : "' +
 str(test_dataset_arn) + '"}]')

        rek_client.distribute_dataset_entries(
            Datasets=datasets
        )

        training_dataset_status, training_dataset_status_message =
 check_dataset_status(
            rek_client, training_dataset_arn)
        test_dataset_status, test_dataset_status_message = check_dataset_status(
            rek_client, test_dataset_arn)

        if training_dataset_status == 'UPDATE_COMPLETE' and test_dataset_status
 == "UPDATE_COMPLETE":
            print("Distribution complete")
        else:
            print("Distribution failed:")
            print(
                f"\ttraining dataset: {training_dataset_status} :
 {training_dataset_status_message}")
            print(
                f"\ttest dataset: {test_dataset_status} :
 {test_dataset_status_message}")

    except ClientError as err:
        logger.exception(
            "Couldn't distribute dataset: %s",err.response['Error']['Message'] )
        raise


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

```
    """

    parser.add_argument(
        "training_dataset_arn", help="The ARN of the training dataset that you
 want to distribute from."
    )

    parser.add_argument(
        "test_dataset_arn", help="The ARN of the test dataset that you want to
 distribute to."
    )


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()

        print(
            f"Distributing training dataset entries
 ({args.training_dataset_arn}) "\
            f"into test dataset ({args.test_dataset_arn}).")

        # Distribute the datasets.

        session = boto3.Session(profile_name='custom-labels-access')
        rekognition_client = session.client("rekognition")

        distribute_dataset_entries(rekognition_client,
                                   args.training_dataset_arn,
                                   args.test_dataset_arn)

        print("Finished distributing datasets.")

    except ClientError as err:
        logger.exception("Problem distributing datasets: %s", err)
        print(f"Problem listing dataset labels: {err}")
```

```
    except Exception as err:
        logger.exception("Problem distributing datasets: %s", err)
        print(f"Problem distributing datasets: {err}")


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

Java V2

使用下列程式碼。提供以下命令行參數：

- 訓練資料集 — 您從中發佈項目之訓練資料集的 ARN。
- test_dataset_arn — 您分發項目的測試資料集的 ARN。

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

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.DatasetDescription;
import software.amazon.awssdk.services.rekognition.model.DatasetStatus;
import software.amazon.awssdk.services.rekognition.model.DescribeDatasetRequest;
import
 software.amazon.awssdk.services.rekognition.model.DescribeDatasetResponse;
import software.amazon.awssdk.services.rekognition.model.DistributeDataset;
import
 software.amazon.awssdk.services.rekognition.model.DistributeDatasetEntriesRequest;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;

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

public class DistributeDatasetEntries {

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

```
   public static DatasetStatus checkDatasetStatus(RekognitionClient rekClient,
String datasetArn)
           throws Exception, RekognitionException {

       boolean distributed = false;
       DatasetStatus status = null;

       // Wait until distribution completes

       do {

           DescribeDatasetRequest describeDatasetRequest =
DescribeDatasetRequest.builder().datasetArn(datasetArn)
                   .build();
           DescribeDatasetResponse describeDatasetResponse =
rekClient.describeDataset(describeDatasetRequest);

           DatasetDescription datasetDescription =
describeDatasetResponse.datasetDescription();

           status = datasetDescription.status();

           logger.log(Level.INFO, " dataset ARN: {0} ", datasetArn);

           switch (status) {

           case UPDATE_COMPLETE:
               logger.log(Level.INFO, "Dataset updated");
               distributed = true;
               break;

           case UPDATE_IN_PROGRESS:
               Thread.sleep(5000);
               break;

           case UPDATE_FAILED:
               String error = "Dataset distribution failed: " +
datasetDescription.statusAsString() + " "
                       + datasetDescription.statusMessage() + " " + datasetArn;
               logger.log(Level.SEVERE, error);
               break;

           default:
```

```
                String unexpectedError = "Unexpected distribution state: " +
datasetDescription.statusAsString() + " "
                        + datasetDescription.statusMessage() + " " + datasetArn;
                logger.log(Level.SEVERE, unexpectedError);

            }

        } while (distributed == false);

        return status;

    }

    public static void distributeMyDatasetEntries(RekognitionClient rekClient,
String trainingDatasetArn,
            String testDatasetArn) throws Exception, RekognitionException {

        try {

            logger.log(Level.INFO, "Distributing {0} dataset to {1} ",
                    new Object[] { trainingDatasetArn, testDatasetArn });

            DistributeDataset distributeTrainingDataset =
DistributeDataset.builder().arn(trainingDatasetArn).build();

            DistributeDataset distributeTestDataset =
DistributeDataset.builder().arn(testDatasetArn).build();

            ArrayList<DistributeDataset> datasets = new ArrayList();

            datasets.add(distributeTrainingDataset);
            datasets.add(distributeTestDataset);

            DistributeDatasetEntriesRequest distributeDatasetEntriesRequest =
DistributeDatasetEntriesRequest.builder()
                    .datasets(datasets).build();

            rekClient.distributeDatasetEntries(distributeDatasetEntriesRequest);

            DatasetStatus trainingStatus = checkDatasetStatus(rekClient,
trainingDatasetArn);
            DatasetStatus testStatus = checkDatasetStatus(rekClient,
testDatasetArn);
```

```
            if (trainingStatus == DatasetStatus.UPDATE_COMPLETE && testStatus ==
DatasetStatus.UPDATE_COMPLETE) {
                logger.log(Level.INFO, "Successfully distributed dataset: {0}",
trainingDatasetArn);

            } else {

                throw new Exception("Failed to distribute dataset: " +
trainingDatasetArn);
            }

        } catch (RekognitionException e) {
            logger.log(Level.SEVERE, "Could not distribute dataset: {0}",
e.getMessage());
            throw e;
        }

    }

    public static void main(String[] args) {

        String trainingDatasetArn = null;
        String testDatasetArn = null;

        final String USAGE = "\n" + "Usage: " + "<training_dataset_arn>
<test_dataset_arn>\n\n" + "Where:\n"
                + "   training_dataset_arn - the ARN of the dataset that you
want to distribute from.\n\n"
                + "   test_dataset_arn - the ARN of the dataset that you want to
distribute to.\n\n";

        if (args.length != 2) {
            System.out.println(USAGE);
            System.exit(1);
        }

        trainingDatasetArn = args[0];
        testDatasetArn = args[1];

        try {

            // Get the Rekognition client.
            RekognitionClient rekClient = RekognitionClient.builder()
```

```
                .credentialsProvider(ProfileCredentialsProvider.create("custom-
labels-access"))
                .region(Region.US_WEST_2)
                .build();

            // Distribute the dataset
            distributeMyDatasetEntries(rekClient, trainingDatasetArn,
 testDatasetArn);

            System.out.println("Datasets distributed.");

            rekClient.close();

        } catch (RekognitionException rekError) {
            logger.log(Level.SEVERE, "Rekognition client error: {0}",
 rekError.getMessage());
            System.exit(1);
        } catch (Exception rekError) {
            logger.log(Level.SEVERE, "Error: {0}", rekError.getMessage());
            System.exit(1);
        }

    }

}
```

刪除資料集

您可以從專案中刪除訓練和測試資料集。

主題

- 刪除資料集 (主控台)
- 刪除 Amazon Rekognition 自訂標籤資料集 (SDK)

刪除資料集 (主控台)

使用下列程序來刪除資料集。之後，如果專案還剩下一個資料集 (訓練或測試)，則會顯示專案詳細資料頁面。如果專案沒有剩餘的資料集，則會顯示「建立資料集」頁面。

如果您刪除訓練資料集，則必須先為專案建立新的訓練資料集，才能訓練模型。如需詳細資訊，請參閱建立包含影像的訓練和測試資料集。

如果您刪除測試資料集，您可以訓練模型，而無需建立新的測試資料集。在訓練期間，會分割訓練資料集，以建立專案的新測試資料集。分割訓練資料集可減少可用於訓練的影像數量。為了維持品質，我們建議您在訓練模型之前先建立新的測試資料集。如需詳細資訊，請參閱將資料集新增至專案。

刪除資料集

1. 開啟亞馬遜重新認知主控台，網址為 https://console.aws.amazon.com/rekognition/。
2. 在左窗格中選擇使用自訂標籤。顯示 Amazon Rekognition 自訂標籤登陸頁面。
3. 在左側導覽窗格中選擇 Projects (專案)。將顯示「專案」檢視。
4. 選擇包含您要刪除之資料集的專案。
5. 在左側導覽窗格中，在專案名稱 (專案名稱) 下，選擇 Datab et
6. 選擇動作
7. 若要刪除訓練資料集，請選擇 [刪除訓練資料集]。
8. 若要刪除測試資料集，請選擇 [刪除測試資料集]。
9. 在 [刪除訓練或測試資料集] 對話方塊中，輸入 delete 以確認您要刪除資料集。
10. 選擇 [刪除訓練] 或 [測試資料集] 以刪除資料集。

刪除 Amazon Rekognition 自訂標籤資料集 (SDK)

資料集的 Amazon Rekognition ource Name (ARN)，請先刪除 Amazon Resource Name (ARN)。DeleteDataset若要取得專案內訓練和測試資料集的 ARN，請呼叫DescribeProjects。該響應包括對ProjectDescription象的數組。資料集 ARN (`DatasetArn`) 和資料集類型 (`DatasetType`) 位於`Datasets`清單中。

如果您刪除訓練資料集，則需要先為專案建立新的訓練資料集，才能訓練模型。如果您刪除測試資料集，您必須先建立新的測試資料集，才能訓練模型。如需詳細資訊，請參閱將資料集新增至專案 (SDK)。

若要刪除資料集 (SDK)

1. 若果您尚未這樣做，請先完成安裝AWS CLI並設定和AWS SDK。如需詳細資訊，請參閱步驟 4：設定 AWS CLI 和 AWS SDKs。
2. 使用下列程式碼來刪除資料集。

AWS CLI

dataset-arn使用您要刪除之資料集的 ARN。

```
aws rekognition delete-dataset --dataset-arn dataset-arn \
  --profile custom-labels-access
```

Python

使用下列程式碼。提供以下命令行參數：

- 資料集-您要刪除之資料集的 ARN。

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

"""
Purpose
Shows how to delete an Amazon Rekognition Custom Labels dataset.
"""
import argparse
import logging
import time
import boto3

from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)


def delete_dataset(rek_client, dataset_arn):
    """
    Deletes an Amazon Rekognition Custom Labels dataset.
    :param rek_client: The Amazon Rekognition Custom Labels Boto3 client.
    :param dataset_arn: The ARN of the dataset that you want to delete.
    """

    try:
        # Delete the dataset,
        logger.info("Deleting dataset: %s", dataset_arn)
```

```
        rek_client.delete_dataset(DatasetArn=dataset_arn)

        deleted = False

        logger.info("waiting for dataset deletion %s", dataset_arn)

        # Dataset might not be deleted yet, so wait.
        while deleted is False:
            try:
                rek_client.describe_dataset(DatasetArn=dataset_arn)
                time.sleep(5)
            except ClientError as err:
                if err.response['Error']['Code'] == 'ResourceNotFoundException':
                    logger.info("dataset deleted: %s", dataset_arn)
                    deleted = True
                else:
                    raise

        logger.info("dataset deleted: %s", dataset_arn)

        return True

    except ClientError as err:
        logger.exception("Couldn't delete dataset - %s: %s",
                         dataset_arn, err.response['Error']['Message'])
        raise


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

    parser.add_argument(
        "dataset_arn", help="The ARN of the dataset that you want to delete."
    )


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()

        print(f"Deleting dataset: {args.dataset_arn}")

        # Delete the dataset.
        session = boto3.Session(profile_name='custom-labels-access')
        rekognition_client = session.client("rekognition")

        delete_dataset(rekognition_client,
                       args.dataset_arn)

        print(f"Finished deleting dataset: {args.dataset_arn}")

    except ClientError as err:
        error_message = f"Problem deleting dataset: {err}"
        logger.exception(error_message)
        print(error_message)


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

Java V2

使用下列程式碼。提供以下命令行參數：

- 資料集-您要刪除之資料集的 ARN。

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

import java.util.logging.Level;
import java.util.logging.Logger;
```

```
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.DeleteDatasetRequest;
import software.amazon.awssdk.services.rekognition.model.DeleteDatasetResponse;
import software.amazon.awssdk.services.rekognition.model.DescribeDatasetRequest;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;

public class DeleteDataset {

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

   public static void deleteMyDataset(RekognitionClient rekClient, String
 datasetArn) throws InterruptedException {

       try {

           logger.log(Level.INFO, "Deleting dataset: {0}", datasetArn);

           // Delete the dataset

           DeleteDatasetRequest deleteDatasetRequest =
 DeleteDatasetRequest.builder().datasetArn(datasetArn).build();

           DeleteDatasetResponse response =
 rekClient.deleteDataset(deleteDatasetRequest);

           // Wait until deletion finishes

           DescribeDatasetRequest describeDatasetRequest =
 DescribeDatasetRequest.builder().datasetArn(datasetArn)
                   .build();

           Boolean deleted = false;

           do {

               try {

                   rekClient.describeDataset(describeDatasetRequest);
                   Thread.sleep(5000);
               } catch (RekognitionException e) {
                   String errorCode = e.awsErrorDetails().errorCode();
```

```
                    if (errorCode.equals("ResourceNotFoundException")) {
                        logger.log(Level.INFO, "Dataset deleted: {0}",
datasetArn);
                        deleted = true;
                    } else {
                        logger.log(Level.SEVERE, "Client error occurred: {0}",
e.getMessage());
                        throw e;
                    }

                }

            } while (Boolean.FALSE.equals(deleted));

            logger.log(Level.INFO, "Dataset deleted: {0} ", datasetArn);

        } catch (

        RekognitionException e) {
            logger.log(Level.SEVERE, "Client error occurred: {0}",
e.getMessage());
            throw e;
        }

    }

    public static void main(String args[]) {

        final String USAGE = "\n" + "Usage: " + "<dataset_arn>\n\n" + "Where:\n"
                + "   dataset_arn - The ARN of the dataset that you want to
delete.\n\n";

        if (args.length != 1) {
            System.out.println(USAGE);
            System.exit(1);
        }

        String datasetArn = args[0];

        try {

            // Get the Rekognition client.
            RekognitionClient rekClient = RekognitionClient.builder()
```

```
                .credentialsProvider(ProfileCredentialsProvider.create("custom-
labels-access"))
                .region(Region.US_WEST_2)
                .build();


            // Delete the dataset
            deleteMyDataset(rekClient, datasetArn);

            System.out.println(String.format("Dataset deleted: %s",
 datasetArn));

            rekClient.close();

        } catch (RekognitionException rekError) {
            logger.log(Level.SEVERE, "Rekognition client error: {0}",
 rekError.getMessage());
            System.exit(1);
        }

        catch (InterruptedException intError) {
            logger.log(Level.SEVERE, "Exception while sleeping: {0}",
 intError.getMessage());
            System.exit(1);
        }

    }

}
```

管理 Amazon Rekognition 自訂標籤模型

Amazon Rekognition 自訂標籤模型是一種數學模型，可預測新影像中物件、場景和概念的存在。它透過在用於訓練模型的影像中尋找圖案來達到此目的。本節說明如何訓練模型、評估其效能以及進行改進。它還向您展示如何使模型可供使用，以及如何在不再需要模型時刪除模型。

主題

- 刪除 Amazon Rekognition 自訂標籤模型
- 為模型加上標籤
- 描述一個模型（SDK）

- 複製 Amazon Rekognition 自訂標籤模型 (SDK)

刪除 Amazon Rekognition 自訂標籤模型

您可以使用 Amazon Rekognition 自訂標籤主控台或使用 DeleteProjectVersionAPI 來刪除模型。如果模型正在執行中或正在訓練，則無法刪除該模型。若要停止執行中的模型，請使用 StopProjectVersionAPI。如需詳細資訊，請參閱停止 Amazon Rekognition 自訂標籤模型 (SDK)。如果模型正在訓練，請等到完成後再刪除模型。

刪除的模型無法取消刪除。

主題

- 刪除 Amazon Rekognition 自訂標籤模型 (主控台)
- 刪除 Amazon Rekognition 自訂標籤模型 (SDK)

刪除 Amazon Rekognition 自訂標籤模型 (主控台)

下列程序說明如何從專案詳細資訊頁面中刪除模型。您也可以從模型的詳細資訊頁面中刪除模型。

若要刪除模型 (主控台)

1. 開啟亞馬遜重新認知主控台，網址為 https://console.aws.amazon.com/rekognition/。
2. 選擇「使用自訂標籤」。
3. 選擇 Get started (開始使用)。
4. 在左側導覽窗格中選擇 Project (專案)。
5. 選擇包含您要刪除之模型的專案。專案詳細資訊頁面隨即開啟。
6. 在「模型」區段中，選取您要刪除的模型。

   > **Note**
   >
   > 如果無法選取模型，表示模型正在執行或正在訓練，且無法刪除。檢查「狀態」(Status) 欄位，並在停止執行中的模型後再試一次，或等到訓練完成。

7. 選擇刪除模型，將顯示刪除模型對話方塊。
8. 輸入刪除以確認刪除。
9. 選擇 Delete (刪除)，刪除模型。刪除模型可能需要一段時間才能完成。

> **Note**
>
> 如果您在刪除模型期間關閉對話方塊，模型仍會被刪除。

刪除 Amazon Rekognition 自訂標籤模型 (SDK)

您可以呼叫DeleteProjectVersion並提供您想要刪除的模型的 Amazon Resource Name (ARN)，以刪除 Amazon Resource Name (ARN)。您可以從 Amazon Rekognition 自訂標籤主控台的模型詳細資料頁面的「使用您的模型」區段取得模型 ARN。或者，調用DescribeProjectVersions並提供以下內容。

- 與模型相關聯的專案 (`ProjectArn`) 的 ARN。
- 模型的版本名稱 (`VersionNames`)。

模型 ARN 是ProjectVersionDescription對象中的`ProjectVersionArn`字段，來自`DescribeProjectVersions`響應。

如果模型正在執行中或正在訓練，則無法刪除該模型。若要判斷模型是否正在執行或訓練，請呼叫DescribeProjectVersions並檢查模型ProjectVersionDescription物件的`Status`欄位。若要停止執行中的模型，請使用 StopProjectVersionAPI。如需詳細資訊，請參閱停止 Amazon Rekognition 自訂標籤模型 (SDK)。您必須等待模型完成訓練，然後才能夠將其刪除。

若要刪除模型 (SDK)

1. 若您尚未這樣做，請安裝AWS CLI並設定和AWS SDK。如需詳細資訊，請參閱步驟 4：設定 AWS CLI 和 AWS SDKs。
2. 使用下列程式碼刪除模型。

   AWS CLI

   `project-version-arn`將的值變更為您想要刪除的專案的名稱。

   ```
   aws rekognition delete-project-version --project-version-arn model_arn \
     --profile custom-labels-access
   ```

   Python

   提供下列命令列參數

- project_arn— 包含您想要刪除的模型的專案的專案的 ARN。
- model_arn-您要刪除之模型版本的 ARN。

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

"""
Purpose
Shows how to delete an existing Amazon Rekognition Custom Labels model.
"""


import argparse
import logging
import time
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)


def find_forward_slash(input_string, n):
    """
    Returns the location of '/' after n number of occurences.
    :param input_string: The string you want to search
    : n: the occurence that you want to find.
    """
    position = input_string.find('/')
    while position >= 0 and n > 1:
        position = input_string.find('/', position + 1)
        n -= 1
    return position


def delete_model(rek_client, project_arn, model_arn):
    """
    Deletes an Amazon Rekognition Custom Labels model.
    :param rek_client: The Amazon Rekognition Custom Labels Boto3 client.
    :param model_arn: The ARN of the model version that you want to delete.
    """
```

```
    try:
        # Delete the model
        logger.info("Deleting dataset: {%s}", model_arn)

        rek_client.delete_project_version(ProjectVersionArn=model_arn)

        # Get the model version name
        start = find_forward_slash(model_arn, 3) + 1
        end = find_forward_slash(model_arn, 4)
        version_name = model_arn[start:end]

        deleted = False

        # model might not be deleted yet, so wait deletion finishes.
        while deleted is False:
            describe_response =
 rek_client.describe_project_versions(ProjectArn=project_arn,

 VersionNames=[version_name])
            if len(describe_response['ProjectVersionDescriptions']) == 0:
                deleted = True
            else:
                logger.info("Waiting for model deletion %s", model_arn)
                time.sleep(5)

        logger.info("model deleted: %s", model_arn)

        return True

    except ClientError as err:
        logger.exception("Couldn't delete model - %s: %s",
                         model_arn, err.response['Error']['Message'])
        raise


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

    parser.add_argument(
        "project_arn", help="The ARN of the project that contains the model that
 you want to delete."
```

```
    )

    parser.add_argument(
        "model_arn", help="The ARN of the model version that you want to
 delete."
    )


def confirm_model_deletion(model_arn):
    """
    Confirms deletion of the model. Returns True if delete entered.
    :param model_arn: The ARN of the model that you want to delete.
    """
    print(f"Are you sure you wany to delete model {model_arn} ?\n", model_arn)

    start = input("Enter delete to delete your model: ")
    if start == "delete":
        return True
    else:
        return 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()

        if confirm_model_deletion(args.model_arn) is True:
            print(f"Deleting model: {args.model_arn}")

            # Delete the model.
            session = boto3.Session(profile_name='custom-labels-access')
            rekognition_client = session.client("rekognition")

            delete_model(rekognition_client,
                         args.project_arn,
                         args.model_arn)
```

```
            print(f"Finished deleting model: {args.model_arn}")
        else:
            print(f"Not deleting model {args.model_arn}")

    except ClientError as err:
        print(f"Problem deleting model: {err}")


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

Java V2

- project_arn— 包含您想要刪除的模型的專案的專案的 ARN。
- model_arn-您想要刪除的模型版本的模型版本的 ARN。

```
//Copyright 2021 Amazon.com, Inc. or its affiliates. All Rights Reserved.
//PDX-License-Identifier: MIT-0 (For details, see https://github.com/
awsdocs/amazon-rekognition-custom-labels-developer-guide/blob/master/LICENSE-
SAMPLECODE.)

import java.net.URI;
import java.util.logging.Level;
import java.util.logging.Logger;

import software.amazon.awssdk.services.rekognition.RekognitionClient;

import
 software.amazon.awssdk.services.rekognition.model.DeleteProjectVersionRequest;
import
 software.amazon.awssdk.services.rekognition.model.DeleteProjectVersionResponse;
import
 software.amazon.awssdk.services.rekognition.model.DescribeProjectVersionsRequest;
import
 software.amazon.awssdk.services.rekognition.model.DescribeProjectVersionsResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;

public class DeleteModel {

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

```
   public static int findForwardSlash(String modelArn, int n) {

       int start = modelArn.indexOf('/');
       while (start >= 0 && n > 1) {
           start = modelArn.indexOf('/', start + 1);
           n -= 1;
       }
       return start;

   }

   public static void deleteMyModel(RekognitionClient rekClient, String
projectArn, String modelArn)
           throws InterruptedException {

       try {

           logger.log(Level.INFO, "Deleting model: {0}", projectArn);

           // Delete the model

           DeleteProjectVersionRequest deleteProjectVersionRequest =
DeleteProjectVersionRequest.builder()
                   .projectVersionArn(modelArn).build();

           DeleteProjectVersionResponse response =
                   rekClient.deleteProjectVersion(deleteProjectVersionRequest);

           logger.log(Level.INFO, "Status: {0}", response.status());

           // Get the model version

           int start = findForwardSlash(modelArn, 3) + 1;
           int end = findForwardSlash(modelArn, 4);

           String versionName = modelArn.substring(start, end);

           Boolean deleted = false;

           DescribeProjectVersionsRequest describeProjectVersionsRequest =
DescribeProjectVersionsRequest.builder()
                   .projectArn(projectArn).versionNames(versionName).build();
```

```
            // Wait until model is deleted.

            do {

                DescribeProjectVersionsResponse describeProjectVersionsResponse
= rekClient

.describeProjectVersions(describeProjectVersionsRequest);

                if
(describeProjectVersionsResponse.projectVersionDescriptions().size()==0) {
                    logger.log(Level.INFO, "Waiting for model deletion: {0}",
modelArn);
                    Thread.sleep(5000);
                } else {
                    deleted = true;
                    logger.log(Level.INFO, "Model deleted: {0}", modelArn);
                }

            } while (Boolean.FALSE.equals(deleted));

            logger.log(Level.INFO, "Model deleted: {0}", modelArn);

        } catch (

        RekognitionException e) {
            logger.log(Level.SEVERE, "Client error occurred: {0}",
e.getMessage());
            throw e;
        }

    }

    public static void main(String args[]) {

        final String USAGE = "\n" + "Usage: " + "<project_arn> <model_arn>\n\n"
+ "Where:\n"
                + "   project_arn - The ARN of the project that contains the
model that you want to delete.\n\n"
                + "   model_version - The ARN of the model that you want to
delete.\n\n";

        if (args.length != 2) {
            System.out.println(USAGE);
```

```
            System.exit(1);
        }

        String projectArn = args[0];
        String modelVersion = args[1];

        try {

            RekognitionClient rekClient = RekognitionClient.builder().build();

            // Delete the model
            deleteMyModel(rekClient, projectArn, modelVersion);

            System.out.println(String.format("model deleted: %s",
 modelVersion));

            rekClient.close();

        } catch (RekognitionException rekError) {
            logger.log(Level.SEVERE, "Rekognition client error: {0}",
 rekError.getMessage());
            System.exit(1);
        }

        catch (InterruptedException intError) {
            logger.log(Level.SEVERE, "Exception while sleeping: {0}",
 intError.getMessage());
            System.exit(1);
        }

    }

}
```

為模型加上標籤

您可以使用標籤來識別、組織、搜尋和篩選 Amazon Rekognition 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，請使用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：

- 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。
- 使用身分和存取權限 (IAM) 控制對模型的存取權限。如需詳細資訊，請參閱使用AWS資源標籤控制對資源的存取權限。
- 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，請參閱執行培訓過的 Amazon Rekognition 自訂標籤模型。

您可以使用 Amazon Rekognition 主控台或使用AWS軟體開發套件來標記模型。

主題

- 標記模型 (主控台)
- 檢視模型標籤
- 標記模型 (SDK)

標記模型 (主控台)

您可以使用 Rekognition 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

此程序說明如何在現有模型中新增標籤或從中移除標籤。您也可以在訓練後將標籤新增到新模型。如需詳細資訊，請參閱訓練 Amazon Rekognition 自訂標籤模型。

使用主控台將標籤新增至現有模型或從中移除標籤

1. 開啟亞馬遜重新認知主控台，網址為 https://console.aws.amazon.com/rekognition/。
2. 選擇 Get started (開始使用)。
3. 在導覽窗格中，選擇 Projects (專案)。
4. 在 Pro jects Resource (Projects) 頁面上，選擇包含您想要您想要標籤的模型的專案。
5. 在導覽窗格中，選擇您先前選擇的專案下，選擇 Models (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 Tags (標籤) 索引標籤。
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. 輸入索引鍵和值。

a. 針對 Key (索引鍵)，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) 若要移除標籤，請在您想要移除的標籤旁邊選擇 Remove (選擇性)。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

您可以使用 Amazon Rekognition 主控台來檢視連接至模型的標籤。

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。

檢視連接至模型的標籤

1. 開啟亞馬遜重新認知主控台，網址為 https://console.aws.amazon.com/rekognition/。
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，選擇您先前選擇的專案下，選擇 Models (模型)。
6. 在 Mo de ations (模型) 部分，選擇您想要檢視其標籤的模型。
7. 在模型的詳細資訊頁面上，選擇 Tags (標籤) 索引標籤。標籤會顯示在「標籤」區段中。

標記模型 (SDK)

您可以使用AWS SDK 來：

- 將標籤新增到新模型
- 將標籤新增到現有模型
- 列出附加到模型的標籤
- 從模型移除標籤

下列AWS CLI範例中的標籤格式如下。

```
--tags '{"key1":"value1","key2":"value2"}'
```

或者，您可以使用此格式。

```
--tags key1=value1,key2=value2
```

如果您尚未安裝AWS CLI，請參閱步驟 4：設定 AWS CLI 和 AWS SDKs。

將標籤新增到新模型

您可以在您建立模型時，將標籤新增到模型中CreateProjectVersion。在Tags陣列輸入參數中指定一或多個標籤。

```
aws rekognition create-project-version --project-arn project arn \
  --version-name version_name \
  --output-config '{ "S3Location": { "Bucket": "output bucket", "Prefix":  "output
 folder" } }' \
  --tags '{"key1":"value1","key2":"value2"}' \
  --profile custom-labels-access
```

如需建立和訓練模型的資訊，請參閱訓練模型 (SDK)。

將標籤新增到現有模型

若要將一或多個標籤新增至現有模型，請使用此TagResource作業。指定要您想要新增的模型的 Amazon Resource Name (ARNResourceArn) (Tags) 和您想要新增的標籤 ()。以下範例將說明如何新增兩個標籤。

```
aws rekognition tag-resource --resource-arn resource-arn \
  --tags '{"key1":"value1","key2":"value2"}' \
  --profile custom-labels-access
```

您可以通過調用來獲取模型的 ARN CreateProjectVersion。

列出模型標籤

若要列出貼附至模型的標籤，請使用ListTagsForResource操作並指定模型的 ARN (ResourceArn)。響應是連接到指定模型標籤鍵和值的映射。

```
aws rekognition list-tags-for-resource --resource-arn resource-arn \
```

```
  --profile custom-labels-access
```

輸出會顯示連接至模型的標籤清單。

```
{
    "Tags": {
        "Dept": "Engineering",
        "Name": "Ana Silva Carolina",
        "Role": "Developer"
    }
}
```

要查看項目中哪些模型具有特定標籤，請調用DescribeProjectVersions以獲取模型列表。然後在響應中調ListTagsForResource用每個模型DescribeProjectVersions。檢查來源的響應ListTagsForResource以查看是否存在所需的標籤。

下面的 Python 3 示例顯示了如何搜索所有項目的特定標籤鍵和值。輸出包括項目 ARN 和模型 ARN，其中一個匹配的密鑰被發現。

若要搜尋標籤值

1. 將下列程式碼儲存為名為的檔案find_tag.py。

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to find a tag value that's associated with models within
your Amazon Rekognition Custom Labels projects.
"""
import logging
import argparse
import boto3

from botocore.exceptions import ClientError


logger = logging.getLogger(__name__)


def find_tag_in_projects(rekognition_client, key, value):
    """
```

```
    Finds Amazon Rekognition Custom Label models tagged with the supplied key and
key value.
    :param rekognition_client: An Amazon Rekognition boto3 client.
    :param key: The tag key to find.
    :param value: The value of the tag that you want to find.
    return: A list of matching model versions (and model projects) that were found.
    """
    try:

        found_tags = []
        found = False

        projects = rekognition_client.describe_projects()
        # Iterate through each project and models within a project.
        for project in projects["ProjectDescriptions"]:
            logger.info("Searching project: %s ...", project["ProjectArn"])

            models = rekognition_client.describe_project_versions(
                ProjectArn=(project["ProjectArn"])
            )

            for model in models["ProjectVersionDescriptions"]:
                logger.info("Searching model %s", model["ProjectVersionArn"])

                tags = rekognition_client.list_tags_for_resource(
                    ResourceArn=model["ProjectVersionArn"]
                )

                logger.info(
                    "\tSearching model: %s for tag: %s value: %s.",
                    model["ProjectVersionArn"],
                    key,
                    value,
                )
                # Check if tag exists.

                if key in tags["Tags"]:
                    if tags["Tags"][key] == value:
                        found = True
                        logger.info(
                            "\t\tMATCH: Project: %s: model version %s",
                            project["ProjectArn"],
                            model["ProjectVersionArn"],
                        )
```

```
                        found_tags.append(
                            {
                                "Project": project["ProjectArn"],
                                "ModelVersion": model["ProjectVersionArn"],
                            }
                        )

        if found is False:
            logger.info("No match for Tag %s with value %s.", key, value)
        return found_tags
    except ClientError as err:
        logger.info("Problem finding tags: %s. ", format(err))
        raise


def main():
    """
    Entry point for example.
    """
    logging.basicConfig(level=logging.INFO,
                        format="%(levelname)s: %(message)s")

    # Set up command line arguments.
    parser = argparse.ArgumentParser(usage=argparse.SUPPRESS)

    parser.add_argument("tag", help="The tag that you want to find.")
    parser.add_argument("value", help="The tag value that you want to find.")

    args = parser.parse_args()
    key = args.tag
    value = args.value

    print(f"Searching your models for tag: {key} with value: {value}.")


    session = boto3.Session(profile_name='custom-labels-access')
    rekognition_client = session.client("rekognition")

    # Get tagged models for all projects.
    tagged_models = find_tag_in_projects(rekognition_client, key, value)

    print("Matched models\n--------------")
    if len(tagged_models) > 0:
        for model in tagged_models:
```

```
            print(
                "Project: {project}\nModel version: {version}\n".format(
                    project=model["Project"], version=model["ModelVersion"]
                )
            )

    else:
        print("No matches found.")

    print("Done.")


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

2. 在命令提示中，輸入以下內容。將#和#替換為您要查找的鍵名和鍵值。

```
python find_tag.py key value
```

從模型刪除標籤

若要移除模型的一或多個標籤，請使用該UntagResource操作。指定要移除的模型 (ResourceArn) 和標籤鍵 (Tag-Keys) 的 ARN。

```
aws rekognition untag-resource --resource-arn resource-arn \
  --tag-keys '["key1","key2"]' \
  --profile custom-labels-access
```

或者，您可以使用此tag-keys格式指定。

```
--tag-keys key1,key2
```

描述一個模型（SDK）

您可以使用DescribeProjectVersions API 來取得模型版本的相關資訊。如果未指定VersionName，則會DescribeProjectVersions傳回專案中所有模型版本的描述。

要描述一個模型（SDK）

1. 若您尚未這樣做，請安裝AWS CLI並設定和AWS SDK。如需詳細資訊，請參閱步驟 4：設定 AWS CLI 和 AWS SDKs。

2. 使用下列範例程式碼來描述模型的版本。

AWS CLI

project-arn將的值變更為您想要描述的專案的 ARN。version-name將的值變更為您想要描述的模型版本。

```
aws rekognition describe-project-versions --project-arn project_arn \
  --version-names version_name \
  --profile custom-labels-access
```

Python

使用下列程式碼。提供以下命令行參數：

- Project_arn — 您要描述之模型的 ARN。
- 模型 _ 版本-您想要您想要描述的模型版本。

例如：python describe_model.py *project_arn model_version*

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

"""
Purpose
Shows how to describe an Amazon Rekognition Custom Labels model.
"""
import argparse
import logging
import boto3

from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)


def describe_model(rek_client, project_arn, version_name):
    """
    Describes an Amazon Rekognition Custom Labels model.
    :param rek_client: The Amazon Rekognition Custom Labels Boto3 client.
    :param project_arn: The ARN of the prject that contains the model.
```

```
    :param version_name: The version name of the model that you want to
 describe.
    """

    try:
        # Describe the model
        logger.info("Describing model: %s for project %s",
                    version_name, project_arn)

        describe_response =
 rek_client.describe_project_versions(ProjectArn=project_arn,

 VersionNames=[version_name])
        for model in describe_response['ProjectVersionDescriptions']:
            print(f"Created: {str(model['CreationTimestamp'])} ")
            print(f"ARN: {str(model['ProjectVersionArn'])} ")
            if 'BillableTrainingTimeInSeconds' in model:
                print(
                    f"Billing training time (minutes):
 {str(model['BillableTrainingTimeInSeconds']/60)} ")
            print("Evaluation results: ")
            if 'EvaluationResult' in model:
                evaluation_results = model['EvaluationResult']
                print(f"\tF1 score: {str(evaluation_results['F1Score'])}")
                print(
                    f"\tSummary location: s3://{evaluation_results['Summary']
['S3Object']['Bucket']}/{evaluation_results['Summary']['S3Object']['Name']}")

            if 'ManifestSummary' in model:
                print(
                    f"Manifest summary location: s3://{model['ManifestSummary']
['S3Object']['Bucket']}/{model['ManifestSummary']['S3Object']['Name']}")
            if 'OutputConfig' in model:
                print(
                    f"Training output location: s3://{model['OutputConfig']
['S3Bucket']}/{model['OutputConfig']['S3KeyPrefix']}")
            if 'MinInferenceUnits' in model:
                print(
                    f"Minimum inference units:
 {str(model['MinInferenceUnits'])}")
            if 'MaxInferenceUnits' in model:
                print(
                    f"Maximum Inference units:
 {str(model['MaxInferenceUnits'])}")
```

```
            print("Status: " + model['Status'])
            print("Message: " + model['StatusMessage'])

    except ClientError as err:
        logger.exception(
            "Couldn't describe model: %s", err.response['Error']['Message'])
        raise


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

    parser.add_argument(
        "project_arn", help="The ARN of the project in which the model resides."
    )
    parser.add_argument(
        "version_name", help="The version of the model that you want to
 describe."
    )


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()

        print(
            f"Describing model: {args.version_name} for project
 {args.project_arn}.")

        # Describe the model.
        session = boto3.Session(profile_name='custom-labels-access')
        rekognition_client = session.client("rekognition")
```

```
        describe_model(rekognition_client, args.project_arn,
                       args.version_name)

        print(
            f"Finished describing model: {args.version_name} for project
 {args.project_arn}.")

    except ClientError as err:
        error_message = f"Problem describing model: {err}"
        logger.exception(error_message)
        print(error_message)
    except Exception as err:
        error_message = f"Problem describing model: {err}"
        logger.exception(error_message)
        print(error_message)


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

Java V2

使用下列程式碼。提供以下命令行參數：

- Project_arn — 您要描述之模型的 ARN。
- 模型 _ 版本-您想要您想要描述的模型版本。

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

package com.example.rekognition;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import
 software.amazon.awssdk.services.rekognition.model.DescribeProjectVersionsRequest;
import
 software.amazon.awssdk.services.rekognition.model.DescribeProjectVersionsResponse;
```

```
import software.amazon.awssdk.services.rekognition.model.EvaluationResult;
import software.amazon.awssdk.services.rekognition.model.GroundTruthManifest;
import software.amazon.awssdk.services.rekognition.model.OutputConfig;
import
 software.amazon.awssdk.services.rekognition.model.ProjectVersionDescription;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;

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

public class DescribeModel {

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

    public static void describeMyModel(RekognitionClient rekClient, String
 projectArn, String versionName) {

        try {

            // If a single version name is supplied, build request argument

            DescribeProjectVersionsRequest describeProjectVersionsRequest =
 null;

            if (versionName == null) {
                describeProjectVersionsRequest =
 DescribeProjectVersionsRequest.builder().projectArn(projectArn)
                        .build();
            } else {
                describeProjectVersionsRequest =
 DescribeProjectVersionsRequest.builder().projectArn(projectArn)
                        .versionNames(versionName).build();
            }

            DescribeProjectVersionsResponse describeProjectVersionsResponse =
 rekClient
                    .describeProjectVersions(describeProjectVersionsRequest);

            for (ProjectVersionDescription projectVersionDescription :
 describeProjectVersionsResponse
                    .projectVersionDescriptions()) {
```

```
                System.out.println("ARN: " +
projectVersionDescription.projectVersionArn());
                System.out.println("Status: " +
projectVersionDescription.statusAsString());
                System.out.println("Message: " +
projectVersionDescription.statusMessage());

                if (projectVersionDescription.billableTrainingTimeInSeconds() !=
null) {
                    System.out.println(
                            "Billable minutes: " +
(projectVersionDescription.billableTrainingTimeInSeconds() / 60));
                }

                if (projectVersionDescription.evaluationResult() != null) {
                    EvaluationResult evaluationResult =
projectVersionDescription.evaluationResult();

                    System.out.println("F1 Score: " +
evaluationResult.f1Score());
                    System.out.println("Summary location: s3://" +
evaluationResult.summary().s3Object().bucket() + "/"
                            + evaluationResult.summary().s3Object().name());
                }

                if (projectVersionDescription.manifestSummary() != null) {
                    GroundTruthManifest manifestSummary =
projectVersionDescription.manifestSummary();
                    System.out.println("Manifest summary location: s3://" +
manifestSummary.s3Object().bucket() + "/"
                            + manifestSummary.s3Object().name());

                }

                if (projectVersionDescription.outputConfig() != null) {
                    OutputConfig outputConfig =
projectVersionDescription.outputConfig();
                    System.out.println(
                            "Training output: s3://" + outputConfig.s3Bucket() +
"/" + outputConfig.s3KeyPrefix());
                }

                if (projectVersionDescription.minInferenceUnits() != null) {
```

```
                    System.out.println("Min inference units: " +
 projectVersionDescription.minInferenceUnits());
                }

                System.out.println();

            }

        } catch (RekognitionException rekError) {
            logger.log(Level.SEVERE, "Rekognition client error: {0}",
 rekError.getMessage());
            throw rekError;
        }

    }

    public static void main(String args[]) {

        String projectArn = null;
        String versionName = null;

        final String USAGE = "\n" + "Usage: " + "<project_arn> <version_name>\n
\n" + "Where:\n"
                + "   project_arn - The ARN of the project that contains the
 models you want to describe.\n\n"
                + "   version_name - (optional) The version name of the model
 that you want to describe. \n\n"
                + "                  If you don't specify a value, all model
 versions are described.\n\n";

        if (args.length > 2 || args.length == 0) {
            System.out.println(USAGE);
            System.exit(1);
        }

        projectArn = args[0];

        if (args.length == 2) {
            versionName = args[1];
        }

        try {

            // Get the Rekognition client.
```

```
            RekognitionClient rekClient = RekognitionClient.builder()
            .credentialsProvider(ProfileCredentialsProvider.create("custom-
labels-access"))
            .region(Region.US_WEST_2)
            .build();

             // Describe the model
            describeMyModel(rekClient, projectArn, versionName);

            rekClient.close();

        } catch (RekognitionException rekError) {
            logger.log(Level.SEVERE, "Rekognition client error: {0}",
 rekError.getMessage());
            System.exit(1);
        }

    }

}
```

複製 Amazon Rekognition 自訂標籤模型 (SDK)

您可以使用此CopyProjectVersion作業將 Amazon Rekognition 自訂標籤模型版本從來源 Amazon Rekognition 自訂標籤專案複製到目的地專案。目標專案可以位於不同的AWS帳戶中，也可以位於同一AWS帳戶中。典型的案例是將已測試的模型從開發AWS帳戶複製到生產AWS帳戶。

或者，您也可以使用來源資料集在目的地帳戶中訓練模型。使用此CopyProjectVersion作業有下列優點。

- 模型行為是一致的。模型訓練是非確定性的，而且使用相同資料集訓練的兩個模型不能保證做出相同的預測。使用複製模型有CopyProjectVersion助於確保複製模型的行為與來源模型一致，而且您不需要重新測試模型。
- 不需要模型訓練。這樣可以為您節省金錢，因為您需要為每次成功訓練模型收取費用。

若要將模型複製到其他AWS帳戶，您必須在目的地AWS帳戶中擁有 Amazon Rekognition 自訂標籤專案。如需有關建立專案的資訊，請參閱建立專案。請務必在目標AWS帳戶中建立專案。

專案原則是以資源為基礎的政策，可為您要複製的模型版本設定複製權限。當目標專案與來源專案的AWS帳戶不同時，您將需要使用專案策略。

在同一帳戶中複製模型版本時，您不需要使用專案策略。但是，如果您想要對這些資源進行更多控制，則可以選擇對帳戶間專案使用專案政策。

您可以呼叫PutProjectPolicy作業，將專案原則附加至來源專案。

您無法使用CopyProjectVersion將模型複製到不同AWS區域中的專案。此外，您無法使用 Amazon Rekognition 自訂標籤主控台複製模型。在這些情況下，您可以使用用來訓練來源模型的資料集來訓練目標專案中的模型。如需詳細資訊，請參閱訓練 Amazon Rekognition 自訂標籤模型。

若要將模型從來源專案複製到目標專案，請執行下列操作：

複製模型

1. 建立專案政策文件。
2. 將專案原則附加至來源專案。
3. 使用CopyProjectVersion操作複製模型。

若要從專案中移除專案原則，請呼叫DeleteProjectPolicy。要獲取附加到項目的項目策略列表，請調用ListProjectPolicies。

主題

- 建立專案政策文件
- 附加專案原則 (SDK)
- 複製模型 (SDK)
- 列出專案政策 (SDK)
- 刪除專案原則 (SDK)

建立專案政策文件

Rekognition 自訂標籤使用以資源為基礎的原則 (稱為專案原則) 來管理模型版本的複製權限。專案政策是 JSON 格式的文件。

專案原則允許或拒絕主體權限，將模型版本從來源專案複製到目的專案。如果目標專案位於不同的AWS帳戶中，則需要專案策略。如果目標專案與來源專案位於相同的AWS帳戶中，且您想要限制對特定模型版本的存取權，也是如此。例如，您可能想要拒絕將許可複製到AWS帳戶中的特定 IAM 角色。

下列範例可讓主參與arn:aws:iam::111111111111:role/Admin者複製模型版本arn:aws:rekognition:us-east-1:123456789012:project/my_project/version/test_1/1627045542080。

```
{
  "Version":"2012-10-17",
  "Statement":[
    {
      "Effect":"Allow",
      "Principal":{
        "AWS":"arn:aws:iam::111111111111:role/Admin"
      },
      "Action":"rekognition:CopyProjectVersion",
      "Resource":"arn:aws:rekognition:us-east-1:111111111111:project/my_project/
version/test_1/1627045542080"
    }
  ]
}
```

> **Note**
>
> Action、ResourcePrincipal、和Effect是專案政策文件中的必填欄位。唯action一支持的是rekognition:CopyProjectVersion。NotActionNotResource、和NotPrincipal是禁止的欄位，且不得出現在專案政策文件中。

如果您未指定專案原則，如果主體具有以身分識別為基礎的原則 (例如，授予呼叫權限)，則與來源專案相同AWS帳戶中的主體仍然可以複製模型CopyProjectVersion。AmazonRekognitionCustomLabelsFullAccess

下列程序會建立可與中的 Python 範例搭配使用的專案政策文件檔案附加專案原則 (SDK)。如果您使用的是put-project-policyAWS CLI命令，則會以 JSON 字串的形式提供專案原則。

若要建立專案政策文件

1. 在文字編輯器中建立下列文件。變更下列值：

   - 效果 — 指定ALLOW以授與複製權限。指定DENY拒絕複製權限。

- 主體 — 您想要允許或拒絕存取您在中指定的模型版本的主參與者Resource。例如，您可以為不同的帳戶指定 AWSAWS 帳戶主體。我們不會限制您可以使用的主參與者。如需詳細資訊，請參閱指定主體。
- Resource Name (ARN)，您想要您想要指定複製許可的模型版本的 Amazon Resource Name (ARN)。如果您想要授與來源專案中所有模型版本的權限，請使用下列格式arn:aws:rekognition:*region*:*account*:project/*source project*/version/*

```
{
  "Version":"2012-10-17",
  "Statement":[
    {
      "Effect":"ALLOW or DENY",
      "Principal":{
        "AWS":"principal"
      },
      "Action":"rekognition:CopyProjectVersion",
      "Resource":"Model version ARN"
    }
  ]
}
```

2. 將專案政策儲存至您的電腦。
3. 依照中的指示，將專案原則附加至來源專案附加專案原則 (SDK)。

附加專案原則 (SDK)

您可以呼叫PutProjectpolicy作業，將專案政策附加到 Amazon Rekognition 自訂標籤專案。

透過呼叫PutProjectPolicy您要新增的每個專案原則，將多個專案原則附加至專案。您最多可以將五個專案專案原則附加至專案。如果您需要您想要附加更多專案政策，請您可以請求提高限制。

當您第一次將唯一的專案原則附加至專案時，請勿在PolicyRevisionId輸入參數中指定修訂 ID。來源的回應PutProjectPolicy是 Amazon Rekognition 自訂標籤為您建立的專案政策修訂 ID。您可以使用修訂 ID 來更新或刪除專案政策的最新修訂版本。Amazon Rekognition 自訂標籤只會保留專案政策的最新修訂版本。如果您嘗試更新或刪除先前的專案原則修訂，就會收到InvalidPolicyRevisionIdException錯誤訊息。

若要更新現有的專案原則，請在PolicyRevisionId輸入參數中指定專案原則的修訂版本識別碼。您可以透過呼叫取得專案中專案原則的修訂版本 ID ListProjectPolicies。

將專案原則貼附至來源專案後，您可以將模型從來源專案複製到目標專案。如需詳細資訊，請參閱複製模型 (SDK)。

若要從專案中移除專案原則，請呼叫DeleteProjectPolicy。要獲取附加到項目的項目策略列表，請調用ListProjectPolicies。

若要將專案原則附加至專案 (SDK)

1. 若您尚未這樣做，請安裝AWS CLI並設定和AWS SDK。如需詳細資訊，請參閱步驟 4：設定 AWS CLI 和 AWS SDKs。
2. 建立專案政策文件。
3. 使用下列程式碼將專案原則附加至包含您要複製之模型版本的信任AWS帳戶中的專案。若要取得專案 ARN，請呼叫DescribeProjects。要獲取模型版本 ARN 調用DescribeProjectVersions。

   AWS CLI

   變更下列值：

   - `project-arn`至包含您要複製之模型版本之信任AWS帳戶中來源專案的 ARN。
   - `policy-name`到您選擇的策略名稱。
   - `principal`要允許或拒絕存取您在中指定之模型版本的主參與者`Model version ARN`。
   - `project-version-arn`到您要複製的模型版本的 ARN。

   如果您要更新現有的專案原則，請指定`policy-revision-id`參數並提供所需專案原則的修訂版本 ID。

   ```
   aws rekognition put-project-policy \
     --project-arn project-arn \
     --policy-name policy-name \
     --policy-document '{ "Version":"2012-10-17", "Statement":
   [{ "Effect":"ALLOW or DENY", "Principal":{ "AWS":"principal" },
    "Action":"rekognition:CopyProjectVersion", "Resource":"project-version-
   arn" }]}' \
     --profile custom-labels-access
   ```

   Python

   使用下列程式碼。提供以下命令行參數：

- project_arn— 您想要您想要您想要您想要您想要您想要您想要您想要您想要您想要連接專案政策
- policy_name— 您選擇的策略名稱。
- project_policy— 包含專案政策文件的檔案。
- policy_revision_id— (選擇性)。如果您想要更新專案原則的現有修訂版本，請指定專案原則的修訂 ID。

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

"""
Purpose
Amazon Rekognition Custom Labels model example used in the service
 documentation:
https://docs.aws.amazon.com/rekognition/latest/customlabels-dg/md-copy-model-
sdk.html
Shows how to attach a project policy to an Amazon Rekognition Custom Labels
 project.
"""

import boto3
import argparse
import logging
import json
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)


def put_project_policy(rek_client, project_arn, policy_name,
 policy_document_file, policy_revision_id=None):
    """
    Attaches a project policy to an Amazon Rekognition Custom Labels project.
    :param rek_client: The Amazon Rekognition Custom Labels Boto3 client.
    :param policy_name: A name for the project policy.
    :param project_arn: The Amazon Resource Name (ARN) of the source project
    that you want to attach the project policy to.
    :param policy_document_file: The JSON project policy document to
    attach to the source project.
```

```
    :param policy_revision_id: (Optional) The revision of an existing policy to
update.
    Pass None to attach new policy.
    :return The revision ID for the project policy.
    """

    try:

        policy_document_json = ""
        response = None

        with open(policy_document_file, 'r') as policy_document:
            policy_document_json = json.dumps(json.load(policy_document))

        logger.info(
            "Attaching %s project_policy to project %s.",
            policy_name, project_arn)

        if policy_revision_id is None:
            response = rek_client.put_project_policy(ProjectArn=project_arn,
                                                     PolicyName=policy_name,

PolicyDocument=policy_document_json)

        else:
            response = rek_client.put_project_policy(ProjectArn=project_arn,
                                                     PolicyName=policy_name,

PolicyDocument=policy_document_json,

PolicyRevisionId=policy_revision_id)

        new_revision_id = response['PolicyRevisionId']

        logger.info(
            "Finished creating project policy %s. Revision ID: %s",
            policy_name, new_revision_id)

        return new_revision_id

    except ClientError as err:
        logger.exception(
            "Couldn't attach %s project policy to project %s: %s }",
            policy_name, project_arn, err.response['Error']['Message'] )
```

```
        raise


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

    parser.add_argument(
        "project_arn",  help="The Amazon Resource Name (ARN) of the project "
        "that you want to attach the project policy to."
    )
    parser.add_argument(
        "policy_name",  help="A name for the project policy."

    )

    parser.add_argument(
        "project_policy",  help="The file containing the project policy JSON"
    )

    parser.add_argument(
        "--policy_revision_id",  help="The revision of an existing policy to
 update. "
        "If you don't supply a value, a new project policy is created.",
        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()

        print(f"Attaching policy to {args.project_arn}")
```

```
        session = boto3.Session(profile_name='custom-labels-access')
        rekognition_client = session.client("rekognition")


        # Attach a new policy or update an existing policy.

        response = put_project_policy(rekognition_client,
                                      args.project_arn,
                                      args.policy_name,
                                      args.project_policy,
                                      args.policy_revision_id)

        print(
            f"project policy {args.policy_name} attached to project
 {args.project_arn}")
        print(f"Revision ID: {response}")

    except ClientError as err:
        print("Problem attaching project policy: %s", err)


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

Java V2

使用下列程式碼。提供以下命令行參數：

- project_arn— 您想要您想要您想要您想要您想要您想要您想要您想要您想要您想要您想要連接專案政策
- project_policy_name— 您選擇的策略名稱。
- project_policy_document— 包含專案政策文件的檔案。
- project_policy_revision_id— (選擇性)。如果您想要更新專案原則的現有修訂版本，請指定專案原則的修訂 ID。

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

```
package com.example.rekognition;

import java.io.IOException;
import java.nio.file.Files;
import java.nio.file.Path;
import java.util.logging.Level;
import java.util.logging.Logger;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import
 software.amazon.awssdk.services.rekognition.model.PutProjectPolicyRequest;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;

public class PutProjectPolicy {

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

    public static void putMyProjectPolicy(RekognitionClient rekClient, String
 projectArn, String projectPolicyName,
             String projectPolicyFileName, String projectPolicyRevisionId)
 throws IOException {

        try {

            Path filePath = Path.of(projectPolicyFileName);



            String policyDocument = Files.readString(filePath);

            String[] logArguments = new String[] { projectPolicyFileName,
 projectPolicyName };

            PutProjectPolicyRequest putProjectPolicyRequest = null;

            logger.log(Level.INFO, "Attaching Project policy: {0} to project:
 {1}", logArguments);

            // Attach the project policy.

            if (projectPolicyRevisionId == null) {
```

```
                putProjectPolicyRequest =
PutProjectPolicyRequest.builder().projectArn(projectArn)

.policyName(projectPolicyName).policyDocument(policyDocument).build();
            } else {
                putProjectPolicyRequest =
PutProjectPolicyRequest.builder().projectArn(projectArn)

.policyName(projectPolicyName).policyRevisionId(projectPolicyRevisionId)
                        .policyDocument(policyDocument)

                        .build();
            }

            rekClient.putProjectPolicy(putProjectPolicyRequest);

            logger.log(Level.INFO, "Attached Project policy: {0} to project:
{1}", logArguments);

        } catch (

        RekognitionException e) {
            logger.log(Level.SEVERE, "Client error occurred: {0}",
e.getMessage());
            throw e;
        }

    }

    public static void main(String args[]) {

        final String USAGE = "\n" + "Usage: "
                + "<project_arn> <project_policy_name> <policy_document>
<project_policy_revision_id>\n\n" + "Where:\n"
                + "   project_arn - The ARN of the project that you want to
attach the project policy to.\n\n"
                + "   project_policy_name - A name for the project policy.\n\n"
                + "   project_policy_document - The file name of the project
policy.\n\n"
                + "   project_policy_revision_id - (Optional) The revision ID of
the project policy that you want to update.\n\n";

        if (args.length < 3 || args.length > 4) {
            System.out.println(USAGE);
```

```
            System.exit(1);
        }

        String projectArn = args[0];
        String projectPolicyName = args[1];
        String projectPolicyDocument = args[2];
        String projectPolicyRevisionId = null;

        if (args.length == 4) {
            projectPolicyRevisionId = args[3];
        }

        try {

            RekognitionClient rekClient = RekognitionClient.builder()
            .credentialsProvider(ProfileCredentialsProvider.create("custom-
labels-access"))
            .region(Region.US_WEST_2)
            .build();


            // Attach the project policy.
            putMyProjectPolicy(rekClient, projectArn, projectPolicyName,
 projectPolicyDocument,
                    projectPolicyRevisionId);

            System.out.println(
                    String.format("project policy %s: attached to project: %s",
 projectPolicyName, projectArn));

            rekClient.close();

        } catch (RekognitionException rekError) {
            logger.log(Level.SEVERE, "Rekognition client error: {0}",
 rekError.getMessage());
            System.exit(1);
        }

        catch (IOException intError) {
            logger.log(Level.SEVERE, "Exception while reading policy document:
 {0}", intError.getMessage());
            System.exit(1);
        }
```

```
    }

}
```

4. 依照中的指示複製模型版本複製模型 (SDK)。

複製模型 (SDK)

您可以使用CopyProjectVersion API 將模型版本從來源專案複製到目標專案。目的地專案可以位於不同的AWS帳戶，但必須是相同的AWS區域。如果目的地專案位於不同的AWS帳戶中 (或者您想要授與AWS帳戶中複製的模型版本的特定權限)，您必須將專案原則附加至來源專案。如需詳細資訊，請參閱建立專案政策文件。CopyProjectVersionAPI 需要存取您的 Amazon S3 儲存貯體。

複製的模型包含來源模型的訓練結果，但不包含來源資料集。

除非您設定適當的權限，否則來源AWS帳戶對複製到目標帳戶的模型沒有所有權。

若要複製模型 (SDK)

1. 若您尚未這樣做，請安裝AWS CLI並設定和AWS SDK。如需詳細資訊，請參閱步驟 4：設定 AWS CLI 和 AWS SDKs。
2. 依照中的指示，將專案原則附加至來源專案附加專案原則 (SDK)。
3. 如果您要將模型複製到其他AWS帳戶，請確定目標AWS科目中有專案。
4. 使用下列程式碼將模型版本複製到目標專案。

   AWS CLI

   變更下列值：

   - source-project-arn至 ARN，其中包含您想要複製的模型版本的來源專案的 ARN。
   - source-project-version-arn至您要複製的模型版本的 ARN。
   - destination-project-arn至 ARN，您想要您想要您想要複製模型的目的地專案的目的地專案的 ARN。
   - version-name至目標專案中模型的版本名稱。
   - bucket至您想要您想要複製來源模型的訓練結果的 S3 儲存貯體。
   - folder到您要將來源模型的訓練結果複製到的資料夾中bucket。
   - (選用)kms-key-id 至模型的 AWS Key Management Service 金鑰 ID。
   - （可選）key到您選擇的標籤鍵。

- (選擇性)value 為您選擇的標籤值。

```
aws rekognition copy-project-version \
  --source-project-arn source-project-arn \
  --source-project-version-arn source-project-version-arn \
  --destination-project-arn destination-project-arn \
  --version-name version-name \
  --output-config '{"S3Bucket":"bucket","S3KeyPrefix":"folder"}' \
  --kms-key-id arn:myKey \
  --tags '{"key":"key"}' \
  --profile custom-labels-access
```

Python

使用下列程式碼。提供以下命令行參數：

- source_project_arn— 來源AWS帳戶中來源專案的 ARN，其中包含您要複製的模型版本。
- source_project_version-arn— 您要複製的來源AWS帳戶中模型版本的 ARN。
- destination_project_arn— 您想要您想要複製模型的目的地專案的 ARN。
- destination_version_name— 目標專案中模型的版本名稱。
- training_results— 您要將來源模型版本的訓練結果複製到的 S3 位置。
- (選用)kms_key_id 至模型的 AWS Key Management Service 金鑰 ID。
- （可選）tag_name到您選擇的標籤鍵。
- (選擇性)tag_value 為您選擇的標籤值。

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

import argparse
import logging
import time
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)
```

```
def copy_model(
    rekognition_client, source_project_arn, source_project_version_arn,
        destination_project_arn, training_results, destination_version_name):
    """
    Copies a version of a Amazon Rekognition Custom Labels model.

    :param rekognition_client: A Boto3 Amazon Rekognition Custom Labels client.
    :param source_project_arn: The ARN of the source project that contains the
    model that you want to copy.
    :param source_project_version_arn: The ARN of the model version that you
 want
    to copy.
    :param destination_project_Arn: The ARN of the project that you want to copy
 the model
    to.
    :param training_results: The Amazon S3 location where training results for
 the model
    should be stored.
    return: The model status and version.
    """
    try:
        logger.info("Copying model...%s from %s to %s ",
 source_project_version_arn,
                    source_project_arn,
                    destination_project_arn)

        output_bucket, output_folder = training_results.replace(
            "s3://", "").split("/", 1)
        output_config = {"S3Bucket": output_bucket,
                         "S3KeyPrefix": output_folder}

        response = rekognition_client.copy_project_version(
            DestinationProjectArn=destination_project_arn,
            OutputConfig=output_config,
            SourceProjectArn=source_project_arn,
            SourceProjectVersionArn=source_project_version_arn,
            VersionName=destination_version_name
        )

        destination_model_arn = response["ProjectVersionArn"]

        logger.info("Destination model ARN: %s", destination_model_arn)
```

```
        # Wait until training completes.
        finished = False
        status = "UNKNOWN"
        while finished is False:
            model_description =
 rekognition_client.describe_project_versions(ProjectArn=destination_project_arn,
                    VersionNames=[destination_version_name])
            status = model_description["ProjectVersionDescriptions"][0]
["Status"]

            if status == "COPYING_IN_PROGRESS":
                logger.info("Model copying in progress...")
                time.sleep(60)
                continue

            if status == "COPYING_COMPLETED":
                logger.info("Model was successfully copied.")

            if status == "COPYING_FAILED":
                logger.info(
                    "Model copy failed: %s ",
                    model_description["ProjectVersionDescriptions"][0]
["StatusMessage"])

            finished = True
    except ClientError:
        logger.exception("Couldn't copy model.")
        raise
    else:
        return destination_model_arn, status


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

    parser.add_argument(
        "source_project_arn",
        help="The ARN of the project that contains the model that you want to
 copy."
    )
```

```
    parser.add_argument(
        "source_project_version_arn",
        help="The ARN of the model version that you want to copy."
    )

    parser.add_argument(
        "destination_project_arn",
        help="The ARN of the project which receives the copied model."
    )

    parser.add_argument(
        "destination_version_name",
        help="The version name for the model in the destination project."
    )

    parser.add_argument(
        "training_results",
        help="The S3 location in the destination account that receives the
 training results for the copied model."
    )


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()

        print(
            f"Copying model version {args.source_project_version_arn} to project
 {args.destination_project_arn}")

        session = boto3.Session(profile_name='custom-labels-access')
        rekognition_client = session.client("rekognition")

        # Copy the model.

        model_arn, status = copy_model(rekognition_client,
```

```
                                        args.source_project_arn,
                                        args.source_project_version_arn,
                                        args.destination_project_arn,
                                        args.training_results,
                                        args.destination_version_name,
                                        )

        print(f"Finished copying model: {model_arn}")
        print(f"Status: {status}")

    except ClientError as err:
        print(f"Problem copying model: {err}")


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

Java V2

使用下列程式碼。提供以下命令行參數：

- source_project_arn— 來源AWS帳戶中來源專案的 ARN，其中包含您要複製的模型版本。
- source_project_version-arn— 您要複製的來源AWS帳戶中模型版本的 ARN。
- destination_project_arn— 您想要您想要複製模型的目的地專案的 ARN。
- destination_version_name— 目標專案中模型的版本名稱。
- output_bucket— 您想要將來源模型版本的訓練結果複製到的 S3 儲存貯體。
- output_folder— 您要將來源模型版本的訓練結果複製到 S3 中的資料夾。

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

package com.example.rekognition;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import
 software.amazon.awssdk.services.rekognition.model.CopyProjectVersionRequest;
```

```
import
 software.amazon.awssdk.services.rekognition.model.CopyProjectVersionResponse;
import
 software.amazon.awssdk.services.rekognition.model.DescribeProjectVersionsRequest;
import
 software.amazon.awssdk.services.rekognition.model.DescribeProjectVersionsResponse;
import software.amazon.awssdk.services.rekognition.model.OutputConfig;
import
 software.amazon.awssdk.services.rekognition.model.ProjectVersionDescription;

import software.amazon.awssdk.services.rekognition.model.RekognitionException;

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

public class CopyModel {

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

    public static ProjectVersionDescription copyMyModel(RekognitionClient
 rekClient,
            String sourceProjectArn,
            String sourceProjectVersionArn,
            String destinationProjectArn,
            String versionName,
            String outputBucket,
            String outputFolder) throws InterruptedException {

        try {

            OutputConfig outputConfig =
 OutputConfig.builder().s3Bucket(outputBucket).s3KeyPrefix(outputFolder).build();

            String[] logArguments = new String[] { versionName,
 sourceProjectArn, destinationProjectArn };

            logger.log(Level.INFO, "Copying model {0} for from project {1} to
 project {2}", logArguments);

            CopyProjectVersionRequest copyProjectVersionRequest =
 CopyProjectVersionRequest.builder()
                    .sourceProjectArn(sourceProjectArn)
                    .sourceProjectVersionArn(sourceProjectVersionArn)
```

```
                    .versionName(versionName)
                    .destinationProjectArn(destinationProjectArn)
                    .outputConfig(outputConfig)
                    .build();

            CopyProjectVersionResponse response =
rekClient.copyProjectVersion(copyProjectVersionRequest);

            logger.log(Level.INFO, "Destination model ARN: {0}",
response.projectVersionArn());
            logger.log(Level.INFO, "Copying model...");

            // wait until copying completes.

            boolean finished = false;

            ProjectVersionDescription copiedModel = null;

            while (Boolean.FALSE.equals(finished)) {
                DescribeProjectVersionsRequest describeProjectVersionsRequest =
DescribeProjectVersionsRequest.builder()
                        .versionNames(versionName)
                        .projectArn(destinationProjectArn)
                        .build();

                DescribeProjectVersionsResponse describeProjectVersionsResponse
= rekClient

.describeProjectVersions(describeProjectVersionsRequest);

                for (ProjectVersionDescription projectVersionDescription :
describeProjectVersionsResponse
                        .projectVersionDescriptions()) {

                    copiedModel = projectVersionDescription;

                    switch (projectVersionDescription.status()) {

                        case COPYING_IN_PROGRESS:
                            logger.log(Level.INFO, "Copying model...");
                            Thread.sleep(5000);
                            continue;

                        case COPYING_COMPLETED:
```

```
                            finished = true;
                            logger.log(Level.INFO, "Copying completed");
                            break;

                        case COPYING_FAILED:
                            finished = true;
                            logger.log(Level.INFO, "Copying failed...");
                            break;

                        default:
                            finished = true;
                            logger.log(Level.INFO, "Unexpected copy status %s",
                                    projectVersionDescription.statusAsString());
                            break;

                    }

                }

            }

            logger.log(Level.INFO, "Finished copying model {0} for from project
{1} to project {2}", logArguments);

            return copiedModel;

        } catch (RekognitionException e) {
            logger.log(Level.SEVERE, "Could not train model: {0}",
e.getMessage());
            throw e;
        }

    }

    public static void main(String args[]) {

        String sourceProjectArn = null;
        String sourceProjectVersionArn = null;
        String destinationProjectArn = null;
        String versionName = null;
        String bucket = null;
        String location = null;

        final String USAGE = "\n" + "Usage: "
```

```
                + "<source_project_arn> <source_project_version_arn>
 <destination_project_arn> <version_name> <output_bucket> <output_folder>\n\n"
                + "Where:\n"
                + "   source_project_arn - The ARN of the project that contains
 the model that you want to copy. \n\n"
                + "   source_project_version_arn - The ARN of the project that
 contains the model that you want to copy. \n\n"
                + "   destination_project_arn - The ARN of the destination
 project that you want to copy the model to. \n\n"
                + "   version_name - A version name for the copied model.\n\n"
                + "   output_bucket - The S3 bucket in which to place the
 training output. \n\n"
                + "   output_folder - The folder within the bucket that the
 training output is stored in. \n\n";

        if (args.length != 6) {
            System.out.println(USAGE);
            System.exit(1);
        }

        sourceProjectArn = args[0];
        sourceProjectVersionArn = args[1];
        destinationProjectArn = args[2];
        versionName = args[3];
        bucket = args[4];
        location = args[5];

        try {

            // Get the Rekognition client.
            RekognitionClient rekClient = RekognitionClient.builder()
            .credentialsProvider(ProfileCredentialsProvider.create("custom-
labels-access"))
            .region(Region.US_WEST_2)
            .build();

            // Copy the model.
            ProjectVersionDescription copiedModel = copyMyModel(rekClient,
                    sourceProjectArn,
                    sourceProjectVersionArn,
                    destinationProjectArn,
                    versionName,
                    bucket,
                    location);
```

```
            System.out.println(String.format("Model copied: %s Status: %s",
                    copiedModel.projectVersionArn(),
                    copiedModel.statusMessage()));

            rekClient.close();

        } catch (RekognitionException rekError) {
            logger.log(Level.SEVERE, "Rekognition client error: {0}",
 rekError.getMessage());
            System.exit(1);
        } catch (InterruptedException intError) {
            logger.log(Level.SEVERE, "Exception while sleeping: {0}",
 intError.getMessage());
            System.exit(1);
        }

    }

}
```

列出專案政策 (SDK)

您可以使用此ListProjectPolicies作業列出附加至 Amazon Rekognition 自訂標籤專案的專案政策。

列出連接至專案 (SDK) 的專案政策

1. 若您尚未這樣做，請安裝AWS CLI並設定和AWS SDK。如需詳細資訊，請參閱步驟 4：設定 AWS CLI 和 AWS SDKs。
2. 使用下面的代碼來列出項目策略。

   AWS CLI

   變更`project-arn`為您要列出附加專案政策之專案的 Amazon 資源名稱。

```
aws rekognition list-project-policies \
  --project-arn project-arn \
  --profile custom-labels-access
```

Python

使用下列程式碼。提供以下命令行參數：

- project_arn — 您要列出附加的專案政策之專案的亞馬遜資源名稱。

例如：python list_project_policies.py *project_arn*

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

"""
Purpose
Amazon Rekognition Custom Labels model example used in the service
 documentation:
https://docs.aws.amazon.com/rekognition/latest/customlabels-dg/md-copy-model-
sdk.html
Shows how to list the project policies in an Amazon Rekogntion Custom Labels
 project.
"""


import argparse
import logging
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)


def display_project_policy(project_policy):
    """
    Displays information about a Custom Labels project policy.
    :param project_policy: The project policy (ProjectPolicy)
    that you want to display information about.
    """
    print(f"Policy name: {(project_policy['PolicyName'])}")
    print(f"Project Arn: {project_policy['ProjectArn']}")
    print(f"Document: {(project_policy['PolicyDocument'])}")
    print(f"Revision ID: {(project_policy['PolicyRevisionId'])}")
    print()
```

```
def list_project_policies(rek_client, project_arn):
    """
    Describes an Amazon Rekognition Custom Labels project, or all projects.
    :param rek_client: The Amazon Rekognition Custom Labels Boto3 client.
    :param project_arn: The Amazon Resource Name of the project you want to use.
    """

    try:

        max_results = 5
        pagination_token = ''
        finished = False

        logger.info("Listing project policies in: %s.", project_arn)
        print('Projects\n--------')
        while not finished:

            response = rek_client.list_project_policies(
                ProjectArn=project_arn, MaxResults=max_results,
 NextToken=pagination_token)

            for project in response['ProjectPolicies']:
                display_project_policy(project)

            if 'NextToken' in response:
                pagination_token = response['NextToken']
            else:
                finished = True

        logger.info("Finished listing project policies.")

    except ClientError as err:
        logger.exception(
            "Couldn't list policies for - %s: %s",
            project_arn,err.response['Error']['Message'])
        raise


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

```
    """

    parser.add_argument(
        "project_arn",  help="The Amazon Resource Name of the project for which
 you want to list project policies."
    )


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()

        print(f"Listing project policies in: {args.project_arn}")

        # List the project policies.

        session = boto3.Session(profile_name='custom-labels-access')
        rekognition_client = session.client("rekognition")

        list_project_policies(rekognition_client,
                              args.project_arn)

    except ClientError as err:
        print(f"Problem list project_policies: {err}")


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

Java V2

使用下列程式碼。提供以下命令行參數：

- project_arn — 專案的 ARN，其中包含您要列出的專案策略。

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

package com.example.rekognition;

import java.util.logging.Level;
import java.util.logging.Logger;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import
 software.amazon.awssdk.services.rekognition.model.ListProjectPoliciesRequest;
import
 software.amazon.awssdk.services.rekognition.model.ListProjectPoliciesResponse;
import software.amazon.awssdk.services.rekognition.model.ProjectPolicy;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;

public class ListProjectPolicies {

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

    public static void listMyProjectPolicies(RekognitionClient rekClient, String
 projectArn) {

        try {

            logger.log(Level.INFO, "Listing project policies for project: {0}",
 projectArn);

            // List the project policies.

            Boolean finished = false;
            String nextToken = null;

            while (Boolean.FALSE.equals(finished)) {

                ListProjectPoliciesRequest listProjectPoliciesRequest =
 ListProjectPoliciesRequest.builder()
                        .maxResults(5)
```

```
                        .projectArn(projectArn)
                        .nextToken(nextToken)
                        .build();

                ListProjectPoliciesResponse response =
rekClient.listProjectPolicies(listProjectPoliciesRequest);

                for (ProjectPolicy projectPolicy : response.projectPolicies()) {

                    System.out.println(String.format("Name: %s",
projectPolicy.policyName()));
                    System.out.println(String.format("Revision ID: %s\n",
projectPolicy.policyRevisionId()));

                }

                nextToken = response.nextToken();

                if (nextToken == null) {
                    finished = true;

                }

            }

            logger.log(Level.INFO, "Finished listing project policies for
project: {0}", projectArn);

        } catch (

        RekognitionException e) {
            logger.log(Level.SEVERE, "Client error occurred: {0}",
e.getMessage());
            throw e;
        }

    }

    public static void main(String args[]) {

        final String USAGE = "\n" + "Usage: " + "<project_arn> \n\n" + "Where:
\n"
                + "   project_arn - The ARN of the project with the project
policies that you want to list.\n\n";
```

```
        ;

        if (args.length != 1) {
            System.out.println(USAGE);
            System.exit(1);
        }

        String projectArn = args[0];

        try {

            RekognitionClient rekClient = RekognitionClient.builder()
            .credentialsProvider(ProfileCredentialsProvider.create("custom-
labels-access"))
            .region(Region.US_WEST_2)
            .build();

            // List the project policies.
            listMyProjectPolicies(rekClient, projectArn);

            rekClient.close();

        } catch (RekognitionException rekError) {
            logger.log(Level.SEVERE, "Rekognition client error: {0}",
 rekError.getMessage());
            System.exit(1);
        }

    }

}
```

刪除專案原則 (SDK)

您可以使用此DeleteProjectPolicy作業從 Amazon Rekognition 自訂標籤專案刪除現有專案政策的修訂。如果您要刪除貼附至專案之專案原則的所有修訂，請使用ListProjectPolicies來取得貼附至專案之每個專案原則的修訂 ID。然後呼叫`DeletePolicy`每個策略名稱。

若要刪除專案原則 (SDK) 的修訂版

1. 若您尚未這樣做，請安裝AWS CLI並設定和AWS SDK。如需詳細資訊，請參閱步驟 4：設定 AWS CLI 和 AWS SDKs。

2. 使用下列程式碼來刪除專案原則。

DeletePolicy 需要ProjectARN，PolicyName和PolicyRevisionId。 ProjectARN並且PolicyName是此 API 所必需的。 PolicyRevisionId是可選的，但可以包含在原子更新的目的。

AWS CLI

變更下列值：

- policy-name至您想要刪除的專案政策的名稱。
- policy-revision-id至您想要刪除的專案政策的修訂 ID。
- project-arn到包含要刪除的項目策略修訂的項目的亞馬遜資源名稱。

```
aws rekognition delete-project-policy \
    --policy-name policy-name \
    --policy-revision-id policy-revision-id \
    --project-arn project-arn \
  --profile custom-labels-access
```

Python

使用下列程式碼。提供以下命令行參數：

- policy-name— 您想要刪除的專案政策的名稱。
- project-arn— 包含要刪除的項目策略的項目的亞馬遜資源名稱。
- policy-revision-id— 您想要刪除的專案政策的修訂 ID。

例如：python delete_project_policy.py *policy_name project_arn####*

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

"""
Purpose
Amazon Rekognition Custom Labels model example used in the service
 documentation:
```

```
https://docs.aws.amazon.com/rekognition/latest/customlabels-dg/md-copy-model-
sdk.html
Shows how to delete a revision of a project policy.
"""

import argparse
import logging
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)


def delete_project_policy(rekognition_client, policy_name,  project_arn,
 policy_revision_id=None):
    """
    Deletes a project policy.

    :param rekognition_client: A Boto3 Amazon Rekognition client.
    :param policy_name: The name of the project policy that you want to delete.
    :param policy_revision_id: The revsion ID for the project policy that you
 want to delete.
    :param project_arn: The Amazon Resource Name of the project that contains
 the project policy
    that you want to delete.
    """
    try:
        logger.info("Deleting project policy: %s", policy_name)

        if policy_revision_id is None:
            rekognition_client.delete_project_policy(
                PolicyName=policy_name,
                ProjectArn=project_arn)

        else:
            rekognition_client.delete_project_policy(
                PolicyName=policy_name,
                PolicyRevisionId=policy_revision_id,
                ProjectArn=project_arn)

        logger.info("Deleted project policy: %s", policy_name)
    except ClientError:
        logger.exception("Couldn't delete project policy.")
        raise
```

```
def confirm_project_policy_deletion(policy_name):
    """
    Confirms deletion of the project policy. Returns True if delete entered.
    :param model_arn: The ARN of the model that you want to delete.
    """
    print(
        f"Are you sure you wany to delete project policy {policy_name} ?\n",
 policy_name)

    delete = input("Enter delete to delete your project policy: ")
    if delete == "delete":
        return True
    else:
        return False


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

    parser.add_argument(
        "policy_name", help="The ARN of the project that contains the project
 policy that you want to delete."
    )

    parser.add_argument(
        "project_arn", help="The ARN of the project project policy you want to
 delete."
    )

    parser.add_argument(
        "--policy_revision_id", help="(Optional) The revision ID of the project
 policy that you want to delete.",
        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()

        if confirm_project_policy_deletion(args.policy_name) is True:
            print(f"Deleting project_policy: {args.policy_name}")

            session = boto3.Session(profile_name='custom-labels-access')
            rekognition_client = session.client("rekognition")

            # Delete the project policy.

            delete_project_policy(rekognition_client,
                                  args.policy_name,
                                  args.project_arn,
                                  args.policy_revision_id)

            print(f"Finished deleting project policy: {args.policy_name}")
        else:
            print(f"Not deleting project policy {args.policy_name}")
    except ClientError as err:
        print(f"Couldn't delete project policy in {args.policy_name}: {err}")



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

Java V2

使用下列程式碼。提供以下命令行參數：

- policy-name— 您想要刪除的專案政策的名稱。
- project-arn— 包含要刪除的項目策略的項目的亞馬遜資源名稱。
- policy-revision-id— 您想要刪除的專案政策的修訂 ID。

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

package com.example.rekognition;

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

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import
 software.amazon.awssdk.services.rekognition.model.DeleteProjectPolicyRequest;

import software.amazon.awssdk.services.rekognition.model.RekognitionException;

public class DeleteProjectPolicy {

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

   public static void deleteMyProjectPolicy(RekognitionClient rekClient, String
 projectArn,
            String projectPolicyName,
            String projectPolicyRevisionId)
            throws InterruptedException {

       try {
            String[] logArguments = new String[] { projectPolicyName,
 projectPolicyRevisionId };

            logger.log(Level.INFO, "Deleting: Project policy: {0} revision:
 {1}", logArguments);

            // Delete the project policy.

            DeleteProjectPolicyRequest deleteProjectPolicyRequest =
 DeleteProjectPolicyRequest.builder()
                    .policyName(projectPolicyName)
                    .policyRevisionId(projectPolicyRevisionId)
```

```
                    .projectArn(projectArn).build();

            rekClient.deleteProjectPolicy(deleteProjectPolicyRequest);

            logger.log(Level.INFO, "Deleted: Project policy: {0} revision: {1}",
 logArguments);

        } catch (

        RekognitionException e) {
            logger.log(Level.SEVERE, "Client error occurred: {0}",
 e.getMessage());
            throw e;
        }

    }

    public static void main(String args[]) {

        final String USAGE = "\n" + "Usage: " + "<project_arn>
 <project_policy_name> <project_policy_revision_id>\n\n"
                + "Where:\n"
                + "   project_arn - The ARN of the project that has the project
 policy that you want to delete.\n\n"
                + "   project_policy_name - The name of the project policy that
 you want to delete.\n\n"
                + "   project_policy_revision_id - The revision of the project
 policy that you want to delete.\n\n";

        if (args.length != 3) {
            System.out.println(USAGE);
            System.exit(1);
        }

        String projectArn = args[0];
        String projectPolicyName = args[1];
        String projectPolicyRevisionId = args[2];

        try {

            RekognitionClient rekClient = RekognitionClient.builder()
            .credentialsProvider(ProfileCredentialsProvider.create("custom-
labels-access"))
            .region(Region.US_WEST_2)
```

```
            .build();

            // Delete the project policy.
            deleteMyProjectPolicy(rekClient, projectArn, projectPolicyName,
 projectPolicyRevisionId);

            System.out.println(String.format("project policy deleted: %s
 revision: %s", projectPolicyName,
                    projectPolicyRevisionId));

            rekClient.close();

        } catch (RekognitionException rekError) {
            logger.log(Level.SEVERE, "Rekognition client error: {0}",
 rekError.getMessage());
            System.exit(1);
        }

        catch (InterruptedException intError) {
            logger.log(Level.SEVERE, "Exception while sleeping: {0}",
 intError.getMessage());
            System.exit(1);
        }

    }

}
```

範例

本節包含可與 Amazon Rekognition 自訂標籤搭配使用的範例相關資訊。

範例	描述
模型回饋方案	示範如何使用人工驗證來改善模型，以建立新的訓練資料集。
亞馬遜自定義標籤演示	顯示呼叫結果的使用者介面示範DetectCustomLabels 。
影片分析	顯示如何使用DetectCustomLabels 從視頻中提取的幀。
分析影像AWS Lambda功能	顯示如何使用DetectCustomLabels 與一個拉姆達函數。
從 CSV 檔案建立清單檔案。	示範如何使用 CSV 檔案建立適合尋找的資訊清單檔案物件、場景和概念與整個圖像（分類）相關聯。

模型回饋方案

「模型意見反應」解決方案可讓您針對模型的預測提供意見反應，並透過人工驗證進行改善。視使用案例而定，只有少數影像的訓練資料集可以成功使用。若要建立更精確的模型，可能需要更大的註解訓練集。使用模型意見反應解決方案，您可以透過模型協助建立更大的資料集。

若要安裝和設定模型回饋解決方案，請參閱〈模型回饋方案。

持續改善模型的工作流程如下：

1. 訓練模型的第一個版本（可能使用小型訓練資料集）。
2. 為模型意見反應解決方案提供未註解的資料集。
3. 模型回饋解決方案使用目前的模型。它啟動人工驗證工作來註釋一個新的數據集。
4. 模型回饋解決方案會根據人為意見反應產生資訊清單檔案，供您用來建立新模型。

亞馬遜自定義標籤演示

Amazon Rekognition 自訂標籤示範展示了一個使用者介面，該介面會使用DetectCustomLabelsAPI。

應用程式會顯示您的 Amazon Rekognition 自訂標籤模型的相關資訊AWS帳戶。選取執行中的模型後，您可以從本端電腦分析影像。如有必要，您可以啟動模型。您也可以停止執行中的模型。該應用程序顯示與其他 AWS 服務（例如亞馬遜 Cognito，亞馬遜 S3 和亞馬遜）的集成CloudFront。

如需詳細資訊，請參閱亞馬遜自定義標籤演示。

影片分析

下面的例子演示了如何使用DetectCustomLabels從視頻中提取的幀。該代碼已在視頻文件中進行了測試MOV和MP4格式。

使用**DetectCustomLabels**與捕獲的幀

1. 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 SDKs。
2. 確保你有rekognition:DetectCustomLabels和AmazonS3ReadOnlyAccess權限。如需詳細資訊，請參閱步驟 4：設定 AWS CLI 和 AWS SDKs。
3. 請使用下列範例程式碼。變更的值videoFile到視頻文件的名稱。變更的值projectVersionArn到您的亞馬遜註冊自定義標籤模型的亞馬遜資源名稱（ARN）。

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

"""
Purpose
Shows how to analyze a local video with an Amazon Rekognition Custom Labels model.
"""
import argparse
import logging
import json
import math
import cv2
import boto3

from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)
```

```
def analyze_video(rek_client, project_version_arn, video_file):
    """
    Analyzes a local video file with an Amazon Rekognition Custom Labels model.
    Creates a results JSON file based on the name of the supplied video file.
    :param rek_client: A Boto3 Amazon Rekognition client.
    :param project_version_arn: The ARN of the Custom Labels model that you want to
 use.
    :param video_file: The video file that you want to analyze.
    """

    custom_labels = []
    cap = cv2.VideoCapture(video_file)
    frame_rate = cap.get(5)  # Frame rate.
    while cap.isOpened():
        frame_id = cap.get(1)  # Current frame number.
        print(f"Processing frame id: {frame_id}")
        ret, frame = cap.read()
        if ret is not True:
            break
        if frame_id % math.floor(frame_rate) == 0:
            has_frame, image_bytes = cv2.imencode(".jpg", frame)

            if has_frame:
                response = rek_client.detect_custom_labels(
                    Image={
                        'Bytes': image_bytes.tobytes(),
                    },
                    ProjectVersionArn=project_version_arn
                )

            for elabel in response["CustomLabels"]:
                elabel["Timestamp"] = (frame_id/frame_rate)*1000
                custom_labels.append(elabel)

    print(custom_labels)

    with open(video_file + ".json", "w", encoding="utf-8") as f:
        f.write(json.dumps(custom_labels))

    cap.release()
```

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

    parser.add_argument(
        "project_version_arn", help="The ARN of the model that you want to use."
    )

    parser.add_argument(
        "video_file", help="The local path to the video that you want to analyze."
    )


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()

        session = boto3.Session(profile_name='custom-labels-access')
        rekognition_client = session.client("rekognition")

        analyze_video(rekognition_client,
                      args.project_version_arn, args.video_file)

    except ClientError as err:
        print(f"Couldn't analyze video: {err}")


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

分析影像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DetectCustomLabels。該功能分析提供的圖像，並返回在圖像中找到的標籤列表。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）

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`DetectCustomLabels`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 函數使用的亞馬遜重新認知自訂標籤模型。
- 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
模型	您希望 Lambda 函數使用的模型的亞馬遜資源名稱（ARN）。您可以從中獲取模型 ARN使用模型Amazon Rekognition 自訂標籤主控台中模型詳細資料頁面的索引標籤。
信心	模型對標籤預測的可信度的最小值 (0—100)。Lambda 函數不會傳回可信度值低於此值的標籤。

8. 選擇儲存以儲存環境變數。
9. 在「」權限窗格，「下」角色名稱，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": "rekognition:DetectCustomLabels",
            "Resource": "*",
            "Effect": "Allow",
            "Sid": "DetectCustomLabels"
        }
    ]
}
```

12. 選擇 下一步。
13. 在政策詳情，輸入策略的名稱，例如DetectCustomLabels-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`DetectCustomLabels`作業包含在預設的 Lambda Python 環境中，作為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新的 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 發行版本新增為函數的一個層。

首先，您要建立包含 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`DetectCustomLabels`並返回在圖像中找到的標籤列表。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 the Amazon Rekognition
Custom Labels model.
"""
import json
import base64
from os import environ
import logging
import boto3

from botocore.exceptions import ClientError

# Set up logging.
logger = logging.getLogger(__name__)

# Get the model ARN and confidence.
model_arn = environ['MODEL_ARN']
min_confidence = int(environ.get('CONFIDENCE', 50))

# Get the boto3 client.
rek_client = boto3.client('rekognition')


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:

        # Determine image source.
        if 'image' in event:
            # Decode the image
            image_bytes = event['image'].encode('utf-8')
            img_b64decoded = base64.b64decode(image_bytes)
            image = {'Bytes': img_b64decoded}


        elif 'S3Object' in event:
            image = {'S3Object':
                     {'Bucket':  event['S3Object']['Bucket'],
```

```
                    'Name': event['S3Object']['Name']}
                   }

        else:
            raise ValueError(
                'Invalid source. Only image base 64 encoded image bytes or S3Object
are supported.')


        # Analyze the image.
        response = rek_client.detect_custom_labels(Image=image,
            MinConfidence=min_confidence,
            ProjectVersionArn=model_arn)

        # Get the custom labels
        labels = response['CustomLabels']

        lambda_response = {
            "statusCode": 200,
            "body": json.dumps(labels)
        }

    except ClientError as err:
        error_message = f"Couldn't analyze image. " + \
            err.response['Error']['Message']

        lambda_response = {
            'statusCode': 400,
            'body': {
                "Error": err.response['Error']['Code'],
                "ErrorMessage": error_message
            }
        }
        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)
            }
        }
```

```
        logger.error("Error function %s: %s",
            context.invoked_function_arn, format(val_error))

    return lambda_response
```

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. 確保你有`lambda:InvokeFunction`權限。您可以使用下列原則。

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

2. 安裝和配置AWS適用於蟒蛇的 SDK。如需詳細資訊，請參閱步驟 4：設定 AWS CLI 和 AWS SDKs。
3. 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）。
4. 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
Test code for running the Amazon Rekognition Custom Labels Lambda
function example code.
"""

import argparse
import logging
import base64
import json
import boto3

from botocore.exceptions import ClientError


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}

    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()

        # Get analysis results.
        result = analyze_image(args.function, args.image)
        status = result['statusCode']

        if status == 200:
            labels = result['body']
            labels = json.loads(labels)
            print(f"There are {len(labels)} labels in the image.")
            for custom_label in labels:
                confidence = int(round(custom_label['Confidence'], 0))
                print(
                    f"Label: {custom_label['Name']}: Confidence: {confidence}%")
        else:
            print(f"Error: {result['statusCode']}")
            print(f"Message: {result['body']}")

    except ClientError as error:
        logging.error(error)
        print(error)


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

5. 執行程式碼。針對命令列引數，提供 Lambda 函數名稱和您要分析的映像檔。您可以提供本機映像的路徑，或提供存放在 Amazon S3 儲存貯體中之映像的 S3 路徑。例如：

```
python client.py function_name s3://bucket/path/image.jpg
```

如果映像位於 Amazon S3 儲存貯體中，請確定它與您在步驟 15 中指定的相同儲存貯體步驟 1：建立AWS Lambda功能（控制台）。

如果成功，則輸出是在圖像中找到的標籤列表。如果沒有傳回標籤，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）。

6. 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。

安全性

您可以保護專案、模型和DetectCustomLabels您的客戶用來偵測自訂標籤的作業。

如需有關保護 Amazon 重新認知的詳細資訊，請參閱亞馬遜重新認知安全。

保護亞馬遜重新認知自訂標籤專案

您可以指定以身分為基礎的政策中指定的資源層級許可，以保護 Amazon Rekognition 自訂標籤專案的安全。如需詳細資訊，請參閱以身分為基礎和以資源為基礎的政策。

您可以保護的 Amazon 重新認知自訂標籤資源包括：

資源	亞馬遜資源名稱格式
Project	ARN: aws: 認知:*: *: 專案/####/日期時間
Model	ARN: aws: 認知:*: *: 專案/####/版本/##/日期時間

下列範例原則顯示如何授與身分識別權限：

- 描述所有項目。
- 建立、啟動、停止和使用特定模型進行推論。
- 建立專案。建立並描述特定模型。
- 拒絕建立特定專案。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllResources",
            "Effect": "Allow",
            "Action": "rekognition:DescribeProjects",
            "Resource": "*"
        },
        {
```

```
            "Sid": "SpecificProjectVersion",
            "Effect": "Allow",
            "Action": [
                "rekognition:StopProjectVersion",
                "rekognition:StartProjectVersion",
                "rekognition:DetectCustomLabels",
                "rekognition:CreateProjectVersion"
            ],
            "Resource": "arn:aws:rekognition:*:*:project/MyProject/version/MyVersion/*"
        },
        {
            "Sid": "SpecificProject",
            "Effect": "Allow",
            "Action": [
                "rekognition:CreateProject",
                "rekognition:DescribeProjectVersions",
                "rekognition:CreateProjectVersion"
            ],
            "Resource": "arn:aws:rekognition:*:*:project/MyProject/*"
        },
        {
            "Sid": "ExplicitDenyCreateProject",
            "Effect": "Deny",
            "Action": [
                "rekognition:CreateProject"
            ],
            "Resource": ["arn:aws:rekognition:*:*:project/SampleProject/*"]
        }
    ]
}
```

安全DetectCustomLabels

用來偵測自訂標籤的身分可能與管理 Amazon Rekognition 自訂標籤模型的身分不同。

您可以安全地訪問身份的訪問DetectCustomLabels藉由將原則套用至身分識別。下列範例會限制存取DetectCustomLabels只和一個特定的模型。身分識別無法存取任何其他 Amazon Rekognition 作業。

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

```
        {
            "Effect": "Allow",
            "Action": [
                "rekognition:DetectCustomLabels"


            ],
            "Resource": "arn:aws:rekognition:*:*:project/MyProject/version/MyVersion/*"
        }
    ]
}
```

AWS 受管政策

我們提供AmazonRekognitionCustomLabelsFullAccess AWS可用於控制對 Amazon Rekognition 自訂標籤的存取權限的受管政策。如需詳細資訊，請參閱AWS 受管政策：AmazonRekognitionCustomLabelsFullAccess。

Amazon Rekognition 自訂標籤中的準則

以下各節提供使用 Amazon Rekognition 自訂標籤時的準則和配額。

支援的區域

如需可以使用 Amazon Rekognition 自訂標籤的AWS區域清單，請參閱 Amazon Web Services 一般參考中的 AWS 區域與端點。

配額

以下是 Amazon Rekognition 自訂標籤的限制清單。如需有關可變更之限制的資訊，請參閱 AWS Service Limits。若要變更限制，請參閱建立案例。

訓練

- 支持的文件格式是 PNG 和 JPEG 圖像格式。
- 模型版本中的訓練資料集數目上限為 1。
- 資料集資訊清單檔案大小上限為 1 GB。
- 每個物件、場景和概念 (分類) 資料集的唯一標籤數目下限為 2。
- 每個物件位置 (偵測) 資料集的唯一標籤數目下限為 1。
- 每個資訊清單的唯一標籤數目上限為 250。
- 每個標籤的最小圖像數量為 1。
- 每個物件位置 (偵測) 資料集的影像數目上限為 250,000。

  亞太區域 (孟買) 和歐洲 (倫敦) AWS 區域的限制為 28,000 張影像。
- 每個物件、場景和概念 (分類) 資料集的影像數目上限為 500,000 個。預設值為 25 萬。若要申請增加，請參閱建立案例。

  亞太區域 (孟買) 和歐洲 (倫敦) AWS 區域的限制為 28,000 張影像。您無法請求提高限制。
- 每張影像的標籤數目上限為 50。
- 影像中邊界方框的最小數目為 0。
- 影像中的邊界方框數目上限為 50。
- Amazon S3 儲存貯體中影像檔案的最小影像尺寸為 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為 4096 像素 x 4096 像素。
- Amazon S3 儲存貯體中影像的檔案大小上限為 15 MB。
- 最大影像畫面比例為 20:1。

測試

- 在一個模型的版本測試數據集的最大數量是 1。
- 資料集資訊清單檔案大小上限為 1 GB。
- 每個物件、場景和概念 (分類) 資料集的唯一標籤數目下限為 2。
- 每個物件位置 (偵測) 資料集的唯一標籤數目下限為 1。
- 每個資料集的唯一標籤數目上限為 250 個。
- 每個標籤的影像數目下限為 0。
- 每個標籤的最大圖像數量為 1000。
- 每個物件位置 (偵測) 資料集的影像數目上限為 250,000。

  亞太區域 (孟買) 和歐洲 (倫敦) AWS 區域的限制為 7,000 張影像。
- 每個物件、場景和概念 (分類) 資料集的影像數目上限為 500,000 個。預設值為 25 萬。若要申請增加，請參閱建立案例。

  亞太區域 (孟買) 和歐洲 (倫敦) AWS 區域的限制為 7,000 張影像。您無法請求提高限制。
- 每個資訊清單每個影像的標籤數目下限為 0。
- 每個資訊清單每個影像的標籤數目上限為 50。
- 每個資訊清單中影像的邊界方塊數目下限為 0。
- 每個資訊清單中影像的邊界方塊數目上限為 50。
- Amazon S3 儲存貯體中影像檔案的最小影像尺寸為 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為 4096 像素 x 4096 像素。
- Amazon S3 儲存貯體中影像的檔案大小上限為 15 MB。
- 支持的文件格式是 PNG 和 JPEG 圖像格式。
- 最大影像畫面比例為 20:1。

偵測

- 以原始位元組傳遞的影像大小上限為 4 MB。

- Amazon S3 儲存貯體中影像的檔案大小上限為 15 MB。
- 輸入影像檔案 (存放在 Amazon S3 儲存貯體或以影像位元組形式提供) 的最小影像尺寸為 64 像素 x 64 像素。
- 輸入影像檔案 (存放在 Amazon S3 中或以影像位元組形式提供) 的最大影像尺寸為 4096 像素 x 4096 像素。
- 支持的文件格式是 PNG 和 JPEG 圖像格式。
- 最大影像畫面比例為 20:1。

模型複製

- 您可以附加至專案的專案原則數目上限為 5。
- 目的地中並行複製工作的數量上限為 5。

Amazon Rekognition 標籤 API 參考

Amazon Rekognition 自訂標籤 API 會記錄為 Amazon Rekognition API 參考內容的一部分。這是 Amazon 認知自訂標籤 API 操作的清單，其中包含適當的 Amazon Rekognition API 參考主題的連結。此外，本文件中的 API 參考連結會移至適當的 Amazon Rekognition 開發人員指南 API 參考主題。如需使用 API 的資訊，請參閱

本節提供工作流程概觀，以便在主控台和AWS SDK 中訓練和使用 Amazon Rekognition 自訂標籤模型。

> **Note**
>
> Amazon Rekognition 自訂標籤現在可以管理專案內的資料集。您可以使用控制台和AWS SDK 為您的項目創建數據集。如果您之前曾使用過 Amazon Rekognition 自訂標籤，您的舊資料集
>
> 可能需要與新專案建立關聯。如需詳細資訊，請參閱 步驟 6：(可選) 將舊資料集與新專案建立關聯

主題

- 決定您的型號類型
- 建立模型
- 改善您的模型
- 啟動模型
- 分析影像
- 停止模型

決定您的型號類型

您首先決定要訓練的模型類型，這取決於您的業務目標。例如，您可以訓練模型以在社交媒體貼文中尋找您的標誌、在商店貨架上識別您的產品，或在裝配線中對機器零件進行分類。

Amazon Rekognition 自訂標籤可訓練以下類型的模型：

- 尋找物件、場景和概念
- 尋找物件位置

- 尋找品牌的位置

為了協助您決定要訓練的模型類型，Amazon Rekognition 自訂標籤提供您可以使用的範例專案。如需詳細資訊，請參閱開始使用亞馬遜自訂標籤。

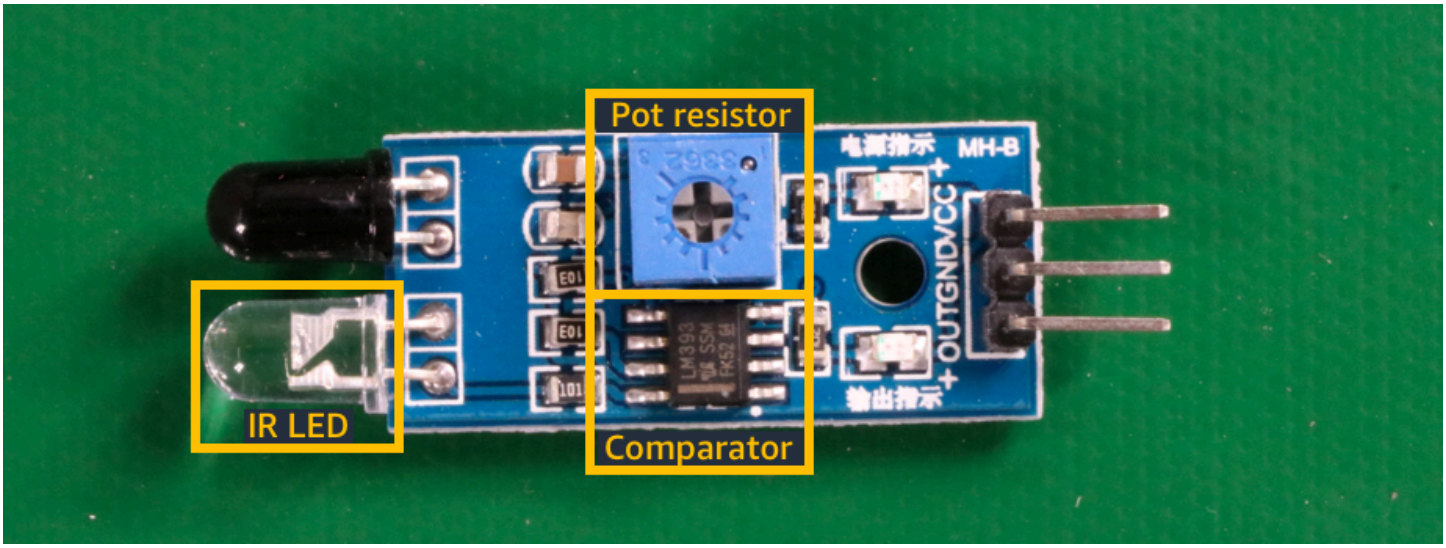
尋找物件、場景和概念

該模型預測與整個圖像相關聯的對象，場景和概念的分類。例如，您可以訓練確定影像是否包含旅遊景點的模型。如需範例專案，請參閱影像分類。

或者，您可以訓練將圖像分類為多個類別的模型。例如，上一個影像可能具有諸如天空顏色、反射或湖泊之類的品類。如需範例專案，請參閱多標籤圖像分類。

尋找物件位置

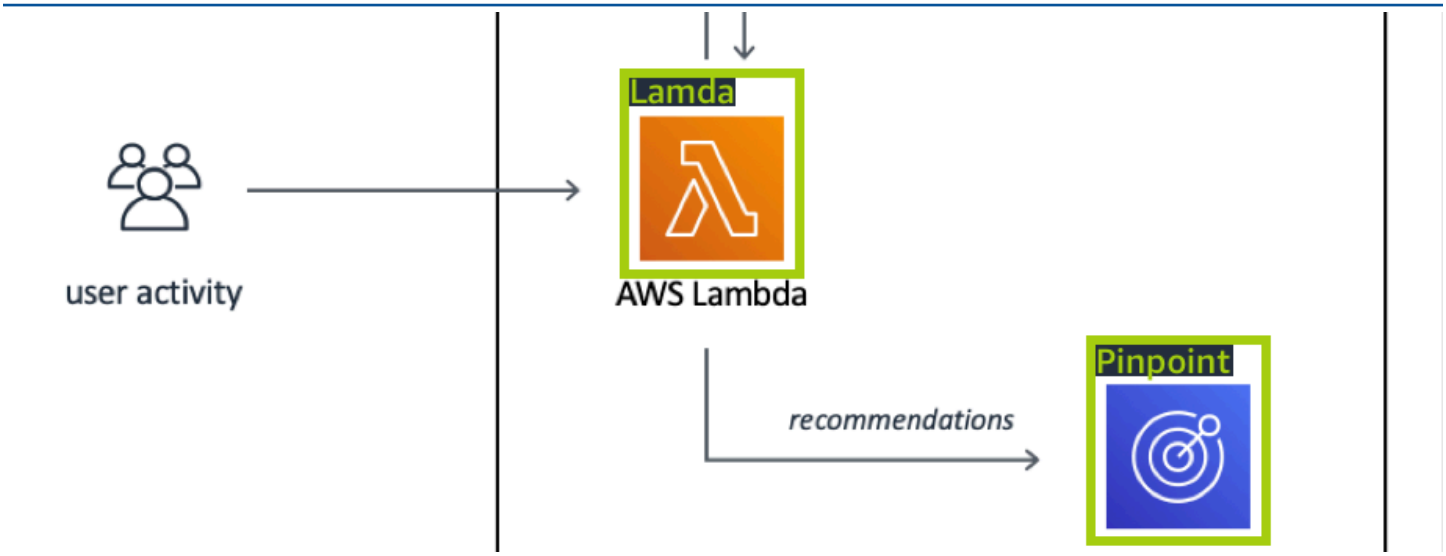
該模型預測圖像上的對象的位置。預測包括物件位置的邊界方框資訊，以及用來識別邊界框內物件的標籤。例如，下圖顯示了電路板各個部分周圍的邊界框，例如比較器或電阻器。



物件本地化範例專案顯示 Amazon Rekognition 自訂標籤如何使用標記的邊界框來訓練尋找物件位置的模型。

尋找品牌的位置

Amazon Rekognition 自訂標籤可訓練在影像上尋找品牌位置 (例如標誌) 的模型。預測包括品牌位置的邊界方框資訊，以及用來識別邊界方框內物件的標籤。如需範例專案，請參閱品牌檢測。



建立模型

建立模型的步驟包括建立專案、建立訓練和測試資料集，以及訓練模型。

建立專案

Amazon Rekognition 自訂標籤專案為建立和管理模型所需的資源群組。專案管理下列項目：

- 資料集 — 用來訓練模型的影像和影像標籤。專案具有訓練資料集和測試資料集。

- 模型 — 您訓練的軟件，以找到您的業務獨特的概念，場景和對象。您可以在專案中擁有多個模型版本。

建議您將專案用於單一使用案例，例如在電路板上尋找電路板零件。

您可以使用 Amazon Rekognition 自訂標籤主控台和 CreateProjectAPI 建立專案。如需詳細資訊，請參閱建立專案。

建立訓練和測試資料集

資料集是描述這些影像的一組影像和標籤。在專案中，您可以建立訓練資料集和測試資料集，Amazon Rekognition 自訂標籤用來訓練和測試模型。

標籤可識別影像中物件周圍的物件、場景、概念或邊界方框。標籤會指定給整個影像 (影像層級)，或是指定給圍繞影像物件的邊界方框。

> **Important**
>
> 如何標記資料集中的影像，會決定 Amazon Rekognition 自訂標籤所建立的模型類型。例如，若要訓練尋找物件、場景和概念的模型，您可以為訓練和測試資料集中的影像指派影像層級標籤。如需詳細資訊，請參閱規劃資料集。

圖像必須為 PNG 和 JPEG 格式，並且您應該遵循輸入的圖像建議。如需詳細資訊，請參閱準備影像。

建立訓練和測試資料集 (主控台)

您可以使用單一資料集，或使用個別的訓練和測試資料集來啟動專案。如果您從單一資料集開始，Amazon Rekognition 自訂標籤會在訓練期間分割您的資料集，以便為您的專案建立訓練資料集 (80%) 和測試資料集 (20%)。如果您希望 Amazon Rekognition 自訂標籤決定要用於訓練和測試的映像，請從單一資料集開始。為了完全控制訓練、測試和效能調整，我們建議您使用個別的訓練和測試資料集來啟動專案。

建立專案的資料集，您可以使用下列其中一種方式匯入影像：

- 從本機電腦匯入影像。

- 從 S3 儲存貯體中匯入映像。Amazon Rekognition 自訂標籤可以使用包含影像的資料夾名稱來標記影像。
- 導入亞馬遜 SageMaker Ground Truth 清單文件。
- 複製現有 Amazon Rekognition 自訂標籤資料集。

如需詳細資訊，請參閱建立包含影像的訓練和測試資料集。

視您匯入影像的位置而定，您的影像可能沒有標記。例如，從本機電腦匯入的影像不會加上標籤。從亞馬遜 SageMaker Ground Truth 清單文件導入的圖像被標記。您可以使用 Amazon Rekognition 自訂標籤主控台來新增、變更和指派標籤。如需詳細資訊，請參閱標記檔案。

若要使用主控台建立訓練和測試資料集，請參閱建立包含影像的訓練和測試資料集。如需包含建立訓練和測試資料集的自學課程，請參閱〈〉教學課程：分類影像。

建立訓練和測試資料集 (SDK)

若要建立訓練和測試資料集，請使用`CreateDataset` API。您可以使用 Amazon Sageemaker 格式資訊清單檔案或複製現有的 Amazon Rekognition 自訂標籤資料集來建立資料集。如需詳細資訊，請參閱建立訓練和測試資料集 (SDK) 如有必要，您可以建立自己的資訊清單檔案。如需詳細資訊，請參閱the section called “建立清單檔案”。

訓練您的模型

使用訓練資料集訓練您的模型。每次訓練模型時，都會建立新版本的模型。在訓練期間，Amazon Rekognition 自訂標籤會測試訓練模型的效能。您可以使用結果來評估和改善模型。訓練需要一段時間才能完成。您只需為成功的模型訓練付費。如需詳細資訊，請參閱訓練 Amazon Rekognition 自訂標籤模型。如果模型訓練失敗，Amazon Rekognition 自訂標籤會提供您可以使用的偵錯資訊。如需詳細資訊，請參閱偵錯失敗的模型訓練。

訓練您的模型（控制台）

若要使用主機訓練模型，請參閱訓練模型（控制台）。

訓練模型 (SDK)

您可以通過調用訓練 Amazon Rekognition 自訂標籤模型CreateProjectVersion。如需詳細資訊，請參閱訓練模型 (SDK)。

改善您的模型

在測試期間，Amazon Rekognition 自訂標籤會建立評估指標，讓您可以使用這些指標來改善訓練過的模型。

評估模型

使用在測試期間建立的效能指標來評估模型的效能。效能指標 (例如 F1、精確度和召回) 可讓您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。如需詳細資訊，請參閱評估模型的指標。

評估模型 (控制台)

檢視效能指標，請參閱存取評估指標 (主控台)。

評估模型 (SDK)

要獲取性能指標，DescribeProjectVersions請致電以獲取測試結果。如需詳細資訊，請參閱存取 Amazon Rekognition 自訂標籤評估指標 (SDK)。測試結果包括控制台中不可用的指標，例如分類結果的混淆矩陣。測試結果以下列格式傳回：

- F1 分數 — 代表模型精確度和召回整體效能的單一值。如需詳細資訊，請參閱F1。

- 摘要檔案位置 — 測試摘要包括整個測試資料集的彙總評估指標，以及每個個別標籤的指標。DescribeProjectVersions傳回摘要檔案的 S3 儲存貯體和資料夾位置。如需詳細資訊，請參閱摘要檔案。
- 評估資訊清單快照位置 — 快照集包含有關測試結果的詳細資料，包括信賴度分級和二進位分類測試的結果，例如誤判。 DescribeProjectVersions傳回快照檔案的 S3 儲存貯體和資料夾位置。如需詳細資訊，請參閱評估資訊清單。

改善您的模型

如果需要改進，您可以新增更多訓練影像或改善資料集標籤。如需詳細資訊，請參閱改善 Amazon Rekognition 自訂標籤模型。您也可以針對模型所做的預測提供意見反應，並使用它來改善模型。如需詳細資訊，請參閱模型回饋方案。

改善您的模型 (控制台)

若要將影像新增至資料集，請參閱將更多影像新增至資料集。若要加入或變更標示，請參閱the section called “標記檔案”。

若要重新訓練模型，請參閱訓練模型（控制台）。

改善您的模型 (SDK)

若要將影像新增至資料集或變更影像的標籤，請使用UpdateDatasetEntries API。UpdateDatasetEntries更新或將 JSON 行添加到清單文件中。每個 JSON 行都包含單一影像的資訊，例如指派的標籤或邊界方框資訊。如需詳細資訊，請參閱新增更多影像 (SDK)。若要檢視資料集中的項目，請使用ListDatasetEntries API。

若要重新訓練模型，請參閱訓練模型 (SDK)。

啟動模型

在您可以使用模型之前，請先使用 Amazon Rekognition 自訂標籤主控台或StartProjectVersion API 啟動模型。您需支付模型執行個體的執行個體。如需詳細資訊，請參閱執行培訓過的模型。

啟動 (主控台)

若要使用主控台啟動模型，請參閱啟動 Amazon Rekognition 自訂標籤模型 (主控台)。

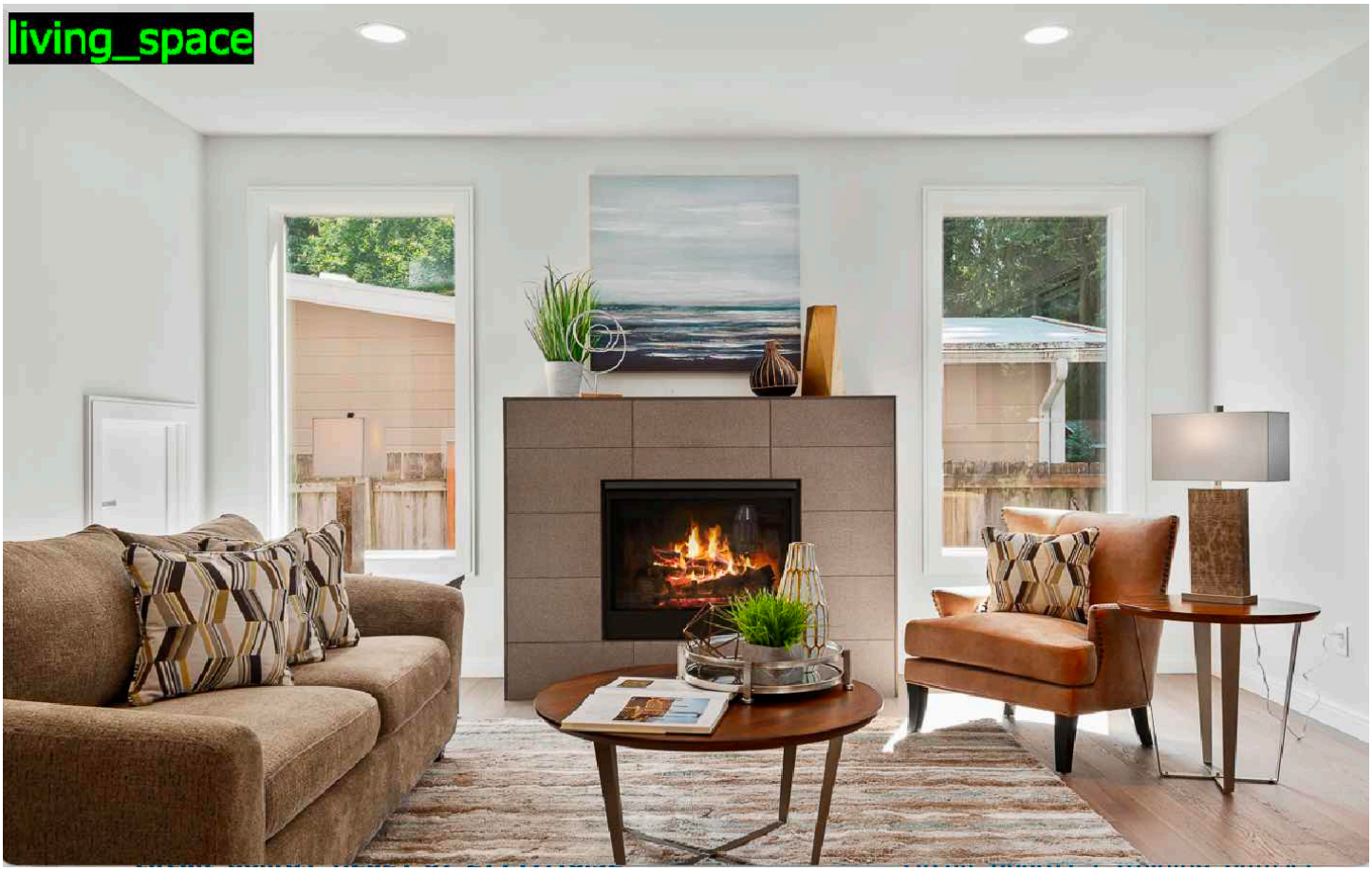
啟動模型

您開始您的模型呼叫StartProjectVersion。如需詳細資訊，請參閱啟動 Amazon Rekognition 自訂標籤模型 (SDK)。

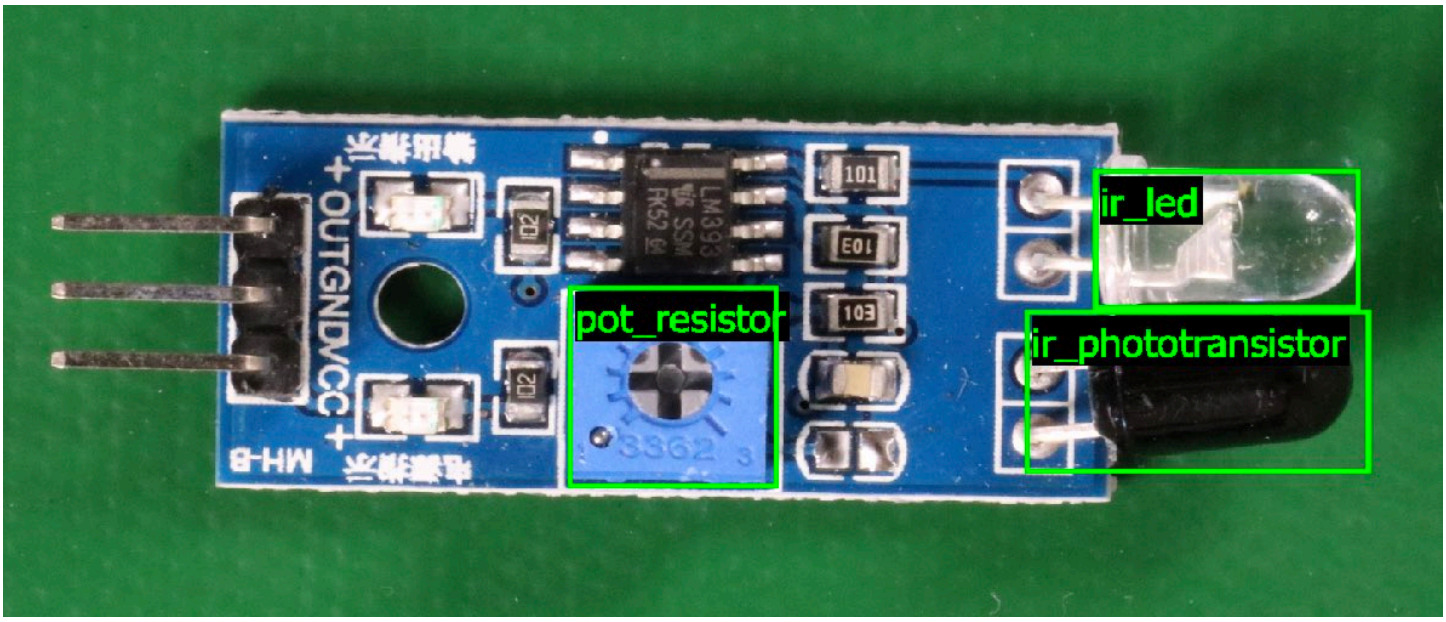
分析影像

若要使用模型分析影像，請使用DetectCustomLabels API。您可以指定本機映像，或存放在 S3 儲存貯體中的映像。此操作還需要您想要使用的模型的 Amazon Resource Name (ARN)。

如果您的模型找到物件、場景和概念，則回應會包含在影像中找到的影像層級標籤清單。例如，下圖展示使用「房間」範例專案中找到的影像層級標籤。



如果模型找到物件位置，則回應會包含影像中已標示的邊界方框清單。邊界方框代表物件在影像上的位置。您可以使用邊界方框資訊，在物件周圍繪製邊界方框。例如，以下影像展示了使用「電路板範例」專案找到的電路板零件周圍的邊界框。



如需詳細資訊，請參閱使用經過培訓的模型分析圖像。

停止模型

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 Amazon Rekognition 自訂標籤主控台或使用StopProjectVersion API 停止模型。如需詳細資訊，請參閱停止 Amazon Rekognition 自訂標籤模型。

停止您的模型（控制台）

若要使用主控台停止執行中的模型，請參閱停止 Amazon Rekognition 自訂標籤模型 (主控台)。

停止您的模型 (SDK)

若要停止執行中的模型，請呼叫StopProjectVersion。如需詳細資訊，請參閱 停止 Amazon Rekognition 自訂標籤模型 (SDK)。

。

訓練您的模型

專案

- CreateProject— 建立 Amazon Rekognition 自訂標籤專案，該專案是資源 (影像、標籤、模型) 和操作 (訓練、評估和偵測) 的邏輯分組。
- DeleteProject— 刪除 Amazon Rekognition 自訂標籤專案
- DescribeProjects— 傳回您所有 Amazon Rekognition 自訂標籤專案的清單。

工程專案

- PutProjectPolicy— 將專案政策附加到信任AWS帳戶中的 Amazon Rekognition 自訂標籤專案。
- ListProjectPolicies— 傳回附加至專案的專案原則清單。
- DeleteProjectPolicy— 刪除現有的專案原則。

資料集

- CreateDataset— 建立 Amazon Rekognition 自訂標籤資料集

- DeleteDataset— 刪除 Amazon Rekognition 自訂標籤資料集。
- DescribeDataset— 描述「Amazon Rekognition 自訂標籤」資料集。
- DistributeDatasetEntries— 將訓練資料集中項目 (影像) 分配至專案的訓練資料集和測試資料集。
- ListDatasetEntries— 傳回 Amazon Rekognition 自訂標籤資料集中的項目 (影像) 清單。
- ListDatasetLabels— 傳回指派給 Amazon Rekognition 自訂標籤資料集的標籤清單。
- UpdateDatasetEntries— 在 Amazon Rekognition 自訂標籤資料集中新增或更新項目 (影像)。

模型

- CreateProjectVersion— 訓練您的「Amazon Rekognition 自訂標籤」模型。
- CopyProjectVersion— 複製您的「Amazon Rekognition 自訂標籤」模型。
- DeleteProjectVersion— 刪除 Amazon Rekognition 自訂標籤模型。
- DescribeProjectVersions— 傳回特定專案中所有 Amazon Rekognition 自訂標籤模型的清單。

Tags (標籤)

- TagResource— 將一個或多個索引鍵值標籤新增至 Amazon Rekognition 自訂標籤模型。
- UntagResource— 移除 Amazon Rekognition 自訂標籤模型中的標籤。

使用您的模型

- DetectCustomLabels— 使用您的自定義標籤模型分析圖像。
- StartProjectVersion— 啟動您的自訂標籤模型。
- StopProjectVersion-停止您的自定義標籤模型。

Amazon Rekognition 自訂標籤的文件歷史記錄

下表說明 Amazon Rekognition 自訂標籤開發人員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 4 月 19 日

變更	描述	日期
已新增模型持續時間	展示如何取得模型所使用的執行時數和推論單位。如需詳細資訊，請參閱報告執行期間和使用的推論單位。	20年 11 月月日
重組資料集內容	將清單文件創建內容移動到清單文件。將資料集轉換主題移至將其他資料集格式轉換為資訊清單檔案。	20年 2 月日
更新適用於的 IAM 指導方針 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 15 日
檢視分類模型的混淆矩陣	Amazon Rekognition 自訂標籤主控台不會顯示分類模型的混淆矩陣。相反地，您可以使用AWS SDK 來取得並顯示混淆矩陣。如需詳細資訊，請參閱檢視模型的混淆矩陣。	2023 年 1 月 4 日
更新 Lambda 函數示例	Lambda 函數範例現在會示範如何分析從本機檔案或 Amazon S3 儲存貯體傳遞的影像。如需詳細資訊，請參閱使用 AWS Lambda 函數分析影像。	2022 年 12 月 2 日

Amazon Rekognition 自訂標籤現在可以複製訓練過的模型	您現在可以將訓練過的模型從一個AWS帳戶複製到同一AWS區域內的另一個AWS帳戶。如需詳細資訊，請參閱複製 Amazon Rekognition 自訂標籤模型 (SDK)。	2022 年 8 月 16 日
Amazon Rekognition 自訂標籤現在可以自動擴展推論單元。	為了協助滿足需求激增，Amazon Rekognition 自訂標籤現在可以調整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 Rekognition 自訂標籤模型。	2022 年 8 月 16 日
從 CSV 檔案建立資訊清單檔案	您現在可以使用從 CSV 檔案讀取影像分類資訊的指令碼來簡化資訊清單檔案的建立。如需詳細資訊，請參閱從 CSV 檔案建立資訊清單檔案。	2022 年 2 月日
Amazon Rekognition 自訂標籤現在可透過專案管理資料集	您可以使用專案來管理用來建立模型的訓練和測試資料集。如需詳細資訊，請參閱瞭解 Amazon Rekognition 自訂標籤。	2021 年 11 月日
Amazon Rekognition 自訂標籤已與AWS CloudFormation	您可以AWS CloudFormation使用 Amazon Rekognition 自訂標籤專案。如需詳細資訊，請參閱使用建立專案AWS CloudFormation。	2021 年 10 月 21 日

更新了入門體驗	Amazon Rekognition 自訂標籤主控台現在包含教學影片和範例專案。如需詳細資訊，請參閱 Amazon Rekognition 自訂標籤入門 Amazon Rekognition 自訂標籤。	2021 年 7 月 22 日
更新臨界值和使用測量結果的相關	有關使用`MinConfidence` input 參數來設定所需臨界值的資訊`DetectCustomLabels` 。如需詳細資訊，請參閱使用訓練過的模型分析影像。	2021 年 6 月 8 日
增加了AWS KMS key支持	您現在可以使用自己的 KMS 金鑰來加密訓練和測試映像。如需詳細資訊，請參閱訓練模型。	2021 年 5 月 19 日
添加標記	Amazon Rekognition 自訂標籤現在支援標記。您可以使用標籤來識別、組織、搜尋和篩選 Amazon Rekognition 自訂標籤模型。如需詳細資訊，請參閱標記模型。	2021 年 3 月 25 日
更新的設定主題	已更新有關如何加密訓練檔案的設定資訊。如需詳細資訊，請參閱步驟 5: (選擇性) 加密訓練檔案。	2021 年 3 月 18 日
新增資料集複製主題	如何將資料集複製到不同AWS區域的相關資訊。如需詳細資訊，請參閱將資料集複製到其他AWS區域。	2021 年 3 月 5 日

添加了亞馬遜 SageMaker GroundTruth 多標籤清單轉換主題	有關如何將 Amazon SageMaker GroundTruth 多標籤格式資訊清單轉換為 Amazon Rekognition 自訂標籤格式資訊清單檔案的資訊。有關詳情，請參閱轉換多標籤 SageMaker Ground Truth 資訊清單檔案。	2021 年 2 月 22 日
已新增模型訓練的除錯資訊	您現在可以使用驗證結果資訊清單來取得有關模型訓練錯誤的深入偵錯資訊。如需詳細資訊，請參閱偵錯失敗模型訓練模型訓練。	2020 年 10 月 8 日
添加了 COCO 轉換信息和示例	如何將 COCO 物件偵測格式資料集轉換為 Amazon Rekognition 自訂標籤資訊清單檔案的相關資訊。如需詳細資訊，請參閱轉換 COCO 資料集。	2020 年 9 月 2 日
Amazon Rekognition 訂標籤現在支援單一物件訓練	若要建立可尋找單一物件位置的 Amazon Rekognition 自訂標籤模型，您現在可以建立只需要一個標籤的資料集。如需詳細資訊，請參閱繪製邊界方框。	2020 年 6 月 25 日
新增專案和模型刪除作業	您現在可以使用主控台和 API 刪除 Amazon Rekognition 自訂標籤專案和模型。如需詳細資訊，請參閱刪除 Amazon Rekognition 自訂標籤模型和刪除 Amazon Rekognition 自訂標籤專案	2020 年 4 月 1 日

增加了 Java 的例子	新增 Java 範例，涵蓋專案建立、模型訓練、模型執行和影像分析。	2019 年 12 月 13 日
新功能和指南	這是 Amazon Rekognition 自訂標籤功能和 Amazon Rekognition 自訂標籤開發人員指南的初始版本。	2019 年 12 月 3 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。