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？ 1
主要優點 1
選擇使用 Amazon Rekognition 自訂標籤 2
Amazon Rekognition Image 標籤偵測 2
Amazon Rekognition 自訂標籤 2
您是第一次使用 Amazon Rekognition 自訂標籤嗎？ 3
設定 Amazon Rekognition 自訂標籤 4
步驟 1：建立 AWS 帳戶 4
註冊 AWS 帳戶 5
程式設計存取權 5
步驟 2：設定主控台權限 6
允許主控台存取 7
存取外部 Amazon S3 儲存貯體 8
指派權限 9
步驟 3：建立主控台儲存貯體 9
步驟 4：設定 AWS CLI 和 AWS SDKs 10
安裝 AWS SDKS 10
授與程式設計存取權 5
設定 SDK 權限 13
呼叫操作 15
步驟 5：(可選) 加密培訓檔案 19
解密使用 加密的檔案 AWS Key Management Service 19
加密複製的培訓和測試圖像 20
步驟 6：(可選) 關聯舊資料集 20
使用舊資料集作為測試資料集 21
了解 Amazon Rekognition 自訂標籤 22
決定模型類型 22
尋找物件、場景和概念 23
尋找物件位置 23
尋找品牌的位置 24
建立模型 24
建立專案 25
建立培訓和測試資料集 25
培訓模型 26

改善模型 27
評估模型 27
改善模型 27
啟動模型 28
啟動模型 (主控台) 28
啟動模型 28
分析圖像 28
停止模型 29
停止模型 (主控台) 29
停止模型 (SDK) 29
開始使用 30
教學課程影片 30
範例專案 30
Image classification 31
多標籤影像分類 31
品牌偵測 32
物件本地化 32
使用範例專案 32
建立範例專案 33
訓練模型 33
使用模型 33
後續步驟 33
步驟 1：選擇範例專案 33
步驟 2：培訓您的模型 36
步驟 3：啟動模型 39
步驟 4：使用模型分析影像 40
取得範例影像 44
步驟 5：停止模型 45
步驟 6：後續步驟 47
分類映像 48
步驟 1：收集您的影像 48
步驟 2：決定課程 49
步驟 3：建立專案 50
步驟 4：建立培訓和測試資料集 51
步驟 5：新增標籤至專案 55
步驟 6：為培訓和測試資料集指派影像層級標籤 55

步驟 7：培訓您的模型 56
步驟 8：啟動模型 60
步驟 9：使用模型分析影像 61
步驟 10：停止模型 64
建立新模型 67
建立專案 67
建立專案 (主控台) 67
建立專案 (SDK) 68
建立專案請求格式 73
建立資料集 73
規劃資料集 74
準備影像 78
建立包含影像的資料集 79
標記檔案 135
偵錯資料集 143
培訓模型 149
培訓模型（主控台） 150
培訓模型 (SDK) 153
偵錯模型訓練 163
終端錯誤 163
非終端 JSON 行驗證錯誤的清單 165
了解清單檔案摘要 166
了解培訓和測試驗證結果清單檔案 170
取得驗證結果 175
修正訓練錯誤 177
終端清單檔案錯誤 179
終端清單檔案內容錯誤 180
非終端 JSON Line 驗證錯誤 189
改善訓練過的模型 212
用於評估模型的指標 212
評估模型效能 212
假設閾值 213
精確度 214
取回 214
F1 214
使用指標 215

存取評估指標 (主控台) 215
存取評估指標 (SDK) 217
存取模型摘要檔案 218
解譯評估資訊清單快照 220
存取摘要檔案和評估清單檔案快照 (SDK) 223
檢視模型的混淆矩陣 224
參考：摘要檔案 231
改善模型 233
資料 233
減少誤報 (精確度更佳) 233
減少漏報 (更佳的取回率) 234
執行培訓過的模型 235
推論單元 235
使用推論單元管理輸送量 236
可用區域 237
啟動模型 238
啟動或停止模型（主控台） 238
啟動模型 (SDK) 239
停止模型 249
停止模型（主控台） 249
停止模型 (SDK) 250
報告持續時間和推論單元 258
使用經過培訓的模型分析圖像 262
偵測自訂標籤操作請求 287
偵測自訂標籤操作回應 288
管理資源 289
管理專案 289
刪除專案 289
描述專案（SDK） 299
使用 建立專案 AWS CloudFormation 305
管理資料集 306
新增資料集 307
新增更多圖像 316
使用現有的資料集建立資料集 (SDK) 325
描述資料集 (SDK) 334
列出資料集條目 (SDK) 339

分配培訓資料集 (SDK) 346
刪除資料集 355
管理模型 362
刪除模型 362
標記模型 371
描述一個模型 (SDK) 378
複製模型 (SDK) 386
自訂標籤範例 421
使用模型意見回饋改善模型 421
Amazon Rekognition 自訂標籤演示 422
在影片中偵測自訂標籤 422
使用 AWS Lambda 函數分析影像 425
步驟 1：建立 AWS Lambda 函數 （主控台） 425
步驟 2：(可選) 建立圖層 (主控台) 427
步驟 3：新增 Python 程式碼 (主控台) 428
步驟 4：嘗試使用您的 Lambda 函數 431
安全 436
保護 Amazon Rekognition 自訂標籤專案的安全 436
保護偵測自訂標籤 437
AWS 受管政策 438
指南和配額 439
支援地區 439
配額 439
培訓 439
測試 440
偵測 440
模型複製 441
API 參考 442
培訓您的模型 450
專案 450
專案政策 450
資料集 450
模型 450
標籤 450
使用您的模型 451
文件歷史記錄 452

..... cdlvii

什麼是 Amazon Rekognition 自訂標籤？

透過 Amazon Rekognition 自訂標籤，您可以識別圖像中特定於您業務需求的物體、標誌和場景。例如，您可以在社交媒體的貼文上找到屬於您的標誌、識別商店貨架上的產品、對生產線中的機器零件進行分類、區分健康植物和受感染植物，或偵側圖片中的動畫人物。

開發自訂模型以分析圖像是一項艱鉅的任務，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工標記的圖像來為模型提供足夠的數據來準確地做出決策。產生這些數據可能需要數月的時間才能收集，而且可能需要大量的標籤員作出準備，將其用於機器學習。

Amazon Rekognition 自訂標籤擴展了 Amazon Rekognition 的現有功能，這些功能已經針對多個類別，數以千萬張圖像進行了培訓。您可以只上傳一小組特定於您的使用案例的培訓圖像（通常是幾百張或更少），而不再需要數千張圖像。您可以使用便利的主控台來完成此操作。如果您的圖像已新增標籤，Amazon Rekognition 自訂標籤可以在短時間內開始培訓模型。如果沒有，您可以直接在標籤界面中標記影像，也可以使用 Amazon SageMaker AI Ground Truth 為您標記影像。

Amazon Rekognition 自訂標籤開始根據您的圖像集進行培訓後，它可以在短短幾小時內為您產生自訂圖像分析模型。在後台，Amazon Rekognition 自訂標籤會自動載入和檢查培訓資料、選擇正確的機器學習演算法、培訓模型並提供模型的效能指標。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當中。

主題

- 主要優點
- 選擇使用 Amazon Rekognition 自訂標籤
- 您是第一次使用 Amazon Rekognition 自訂標籤嗎？

主要優點

簡化的資料標記

Amazon Rekognition 自訂標籤主控台提供了一個可視化介面，可以快速、簡單地為圖像添加標籤。該界面可讓您將標籤應用於整個圖像。您也可以透過點擊並拖曳介面使用邊界框來識別和標記圖像中的特定物體。或者，如果您擁有大型資料集，則可以使用 Amazon SageMaker Ground Truth 有效地擴展標記圖像。

自動化機器學習

建立自訂模型不需要機器學習專業知識。Amazon Rekognition 自訂標籤包含自動化機器學習 (AutoML) 功能，可為您處理好機器學習。提供培訓圖像後，Amazon Rekognition 自訂標籤可以自動載入和檢查資料、選擇正確的機器學習演算法、培訓模型並提供模型效能指標。

簡化的模型評估、推理和反饋

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，您可以看到模型預測與分配的標籤的並排比較。您也可以檢閱詳細的效能指標，例如精確度、召回、F1 分數和信賴度分數。您可以立即開始使用模型進行圖像分析，也可以使用更多圖像重演和重新培訓新版本以提高效能。開始使用模型後，您可以追蹤預測、修正任何錯誤，並使用反饋資料重新培訓新模型版本並提高效能。

選擇使用 Amazon Rekognition 自訂標籤

Amazon Rekognition 提供兩種特徵，可讓您用來尋找圖像中的標籤 (物體、場景和概念)：Amazon Rekognition 自訂標籤 和 Amazon Rekognition Image 標籤偵測。使用以下資訊來確定您應該使用哪種特徵。

Amazon Rekognition Image 標籤偵測

您可以使用 Amazon Rekognition Image 中的標籤偵測特徵，擴展識別、分類和搜尋圖像和影片中的常見 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，例如汽車和貨車、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 Image 標籤偵測，因為您不需要培訓模型。若要取得 Amazon Rekognition Image 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 Image 標籤偵測找不到的標籤，例如組裝線上的自訂機器零件，我們建議您使用 Amazon Rekognition 自訂標籤。

Amazon Rekognition 自訂標籤

您可以使用 Amazon Rekognition 自訂標籤輕鬆培訓機器學習模型，在圖像中尋找符合您業務需求的特有標籤 (物體、標誌、場景和概念)。

Amazon Rekognition 自訂標籤可以將圖像分類 (圖像層級預測) 或偵測圖像中的物體位置 (物體/邊界框層級預測)。

Amazon Rekognition 自訂標籤為您可以偵測的物體和場景類型提供了更大的靈活性。例如，您可以使用 Amazon Rekognition Image 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 自訂標籤的範例。

- 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
- 在零售貨架上尋找特定品牌的產品
- 將農產品的品質進行分類（例如腐爛、成熟或未經烹調的）

> **Note**
>
> Amazon Rekognition 自訂標籤不適用於分析臉孔、偵測文字或尋找圖像中不安全的圖像內容。若要執行這些任務，您可以使用 Amazon Rekognition Image。如需更多詳細資訊，請參閱 什麼是 Amazon Rekognition 自訂標籤。

您是第一次使用 Amazon Rekognition 自訂標籤嗎？

如果您是 Amazon Rekognition 自訂標籤的首次用戶，我們建議您按順序閱讀以下部分：

1. 設定 Amazon Rekognition 自訂標籤 — 本節將會說明如何設定您的帳戶資料。
2. 了解 Amazon Rekognition 自訂標籤 – 本節將會說明建立模型的工作流程。
3. Amazon Rekognition 自訂標籤入門 – 在本節中，您將會使用 Amazon Rekognition 自訂標籤建立的範例專案來訓練模型。
4. 分類映像 – 本節將會說明如何使用您建立的資料集來訓練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
- 步驟 2：設定 Amazon Rekognition 自訂標籤主控台權限
- 步驟 3：建立主控台儲存貯體
- 步驟 4：設定 AWS CLI 和 AWS SDKs
- 步驟 5：(可選) 加密培訓檔案
- 步驟 6：(可選) 將舊資料集與新專案建立關聯

步驟 1：建立 AWS 帳戶

在此步驟中，您將建立 AWS 帳戶、建立 管理使用者，以及了解如何授予 AWS SDK 的程式設計存取權。

主題

- 註冊 AWS 帳戶
- 程式設計存取權

註冊 AWS 帳戶

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

程式設計存取權

如果使用者想要與 AWS 外部互動，則需要程式設計存取 AWS 管理主控台。授予程式設計存取權的方式取決於正在存取的使用者類型 AWS。

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	根據
IAM	（建議）使用主控台登入資料做為臨時登入資料，以簽署對 AWS CLI、 AWS SDKs程式設計請求。 AWS APIs	請依照您要使用的介面所提供的指示操作。 • 如需 AWS CLI，請參閱AWS Command Line Interface 《使用者指南》中的登入以進行 AWS 本機開發。 • AWS SDKs，請參閱 AWS SDKs 和工具參考指南中的登入以進行 AWS 本機開發。
人力資源身分 (IAM Identity Center 中管理的使用者)	使用暫時登入資料簽署對 AWS CLI、 AWS SDKs程式設計請求。 AWS APIs	請依照您要使用的介面所提供的指示操作。 • 如需 AWS CLI，請參閱AWS Command Line Interface 《使用者指南》中的設定 AWS CLI 要使用 AWS IAM Identity Center的 。 • AWS SDKs、工具和 AWS APIs，請參閱 AWS SDK 和

哪個使用者需要程式設計存取權？	到	根據
		工具參考指南中的 SDKsIAM Identity Center 身分驗證。
IAM	使用暫時登入資料簽署對 AWS CLI、 AWS SDKs程式設計請求。 AWS APIs	遵循《IAM 使用者指南》中將臨時登入資料與 AWS 資源搭配使用的指示。
IAM	(不建議使用) 使用長期憑證簽署對 AWS CLI、 AWS SDKs程式設計請求。 AWS APIs	請依照您要使用的介面所提供的指示操作。 • 如需 AWS CLI，請參閱AWS Command Line Interface 《 使用者指南》中的使用 IAM 使用者憑證進行身分驗證。 • AWS SDKs和工具，請參閱 AWS SDKs和工具參考指南中的使用長期憑證進行身分驗證。 • 對於 AWS APIs，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、SageMaker AI Ground 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。

JSON

```
{
    "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 amzn-s3-demo-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:::amzn-s3-demo-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：

- 中的使用者和群組 AWS IAM Identity Center：

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

  建立聯合身分的角色。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。請按照《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。您應該在稍後可以參考的位置記下主控台儲存貯體名稱，因為您可能需要在 AWS SDK 操作或主控台任務中使用儲存貯體名稱，例如建立資料集。

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 管理主控台 ，並在 https：//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，請下載您正在使用的程式設計語言 AWS 開發套件。

主題

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

安裝 AWS SDKS

依照步驟下載和設定 AWS SDKs。

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 AWS CLI、Java 和 Python 的範例。如需安裝 AWS SDKs的詳細資訊，請參閱適用於 Amazon Web Services 的工具。

授與程式設計存取權

您可以在本機電腦或其他 AWS 環境上執行本指南中的 AWS CLI 和 程式碼範例，例如 Amazon Elastic Compute Cloud 執行個體。若要執行範例，您需要授予範例使用的 AWS SDK 操作存取權。

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，建議您使用短期登入資料來授予使用者 AWS SDK 操作的存取權。如需在本機電腦上執行 AWS CLI 和 程式碼範例的特定資訊，請參閱 在本機電腦上使用設定檔。

如果使用者想要與 AWS 外部互動，則需要程式設計存取 AWS 管理主控台。授予程式設計存取權的方式取決於正在存取的使用者類型 AWS。

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	根據
IAM	（建議）使用主控台登入資料做為臨時登入資料，以簽署對 AWS CLI、 AWS SDKs程式設計請求。 AWS APIs	請依照您要使用的介面所提供的指示操作。 • 如需 AWS CLI，請參閱AWS Command Line Interface 《使用者指南》中的登入以進行 AWS 本機開發。 • AWS SDKs，請參閱 AWS SDKs 和工具參考指南中的登入以進行 AWS 本機開發。
人力資源身分 (IAM Identity Center 中管理的使用者)	使用暫時登入資料簽署對 AWS CLI、 AWS SDKs程式設計請求。 AWS APIs	請依照您要使用的介面所提供的指示操作。 • 如需 AWS CLI，請參閱AWS Command Line Interface 《使用者指南》中的設定 AWS CLI 要使用的 AWS IAM Identity Center。 • AWS SDKs、工具和 AWS APIs，請參閱 AWS SDK 和工具參考指南中的 SDKsIAM Identity Center 身分驗證。

哪個使用者需要程式設計存取權？	到	根據
IAM	使用暫時登入資料簽署對 AWS CLI、 AWS SDKs程式設計請求。 AWS APIs	遵循《IAM 使用者指南》中將臨時登入資料與 AWS 資源搭配使用的指示。
IAM	(不建議使用) 使用長期憑證簽署對 AWS CLI、 AWS SDKs程式設計請求。 AWS APIs	請依照您要使用的介面所提供的指示操作。 • 如需 AWS CLI，請參閱AWS Command Line Interface 《 使用者指南》中的使用 IAM 使用者憑證進行身分驗證。 • AWS SDKs和工具，請參閱 AWS SDKs和工具參考指南中的使用長期憑證進行身分驗證。 • 對於 AWS APIs，請參閱《 IAM 使用者指南》中的管理 IAM 使用者的存取金鑰。

在本機電腦上使用設定檔

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

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

描述檔代表的使用者必須具有呼叫 Amazon Rekognition 自訂標籤 SDK 操作和範例所需其他 AWS SDK 操作的許可。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。

若要建立使用 AWS CLI 和 程式碼範例的設定檔，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 管理的使用者 AWS IAM Identity Center) — 遵循設定要使用的 AWS CLI AWS IAM Identity Center中的指示。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

您不應使用使用者登入資料在 AWS 環境中簽署 AWS SDK 呼叫，例如在 AWS Lambda 函數中執行的生產程式碼。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 — 使用 Lambda 在擔任 Lambda 函數的執行角色時自動提供給函數的臨時登入資料。這些憑證可在 Lambda 環境變數中使用。您不需要指定設定檔。如需更多詳細資訊，請參閱 Lambda 執行角色。
- Amazon EC2 — 使用 Amazon EC2 執行個體中繼資料端點憑證提供者。提供者會使用您附加到 Amazon EC2 執行個體的 Amazon EC2 執行個體設定檔，為您自動產生和重新整理憑證。如需更多詳細資訊，請參閱 使用 IAM 角色向在 Amazon EC2 執行個體上執行的應用程式授予權限。
- Amazon Elastic Container Service — 使用容器憑證提供者。Amazon ECS 會將憑證傳送並重新整理到中繼資料端點。您指定的 任務 IAM 角色 提供了用於管理應用程式使用的憑證的策略。如需更多詳細資訊，請參閱 與 AWS 服務互動。

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，要呼叫 檢測自訂標籤，您需要執行 `rekognition:DetectCustomLabels` 的權限。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 受管政策來完整存取 Amazon Rekognition 自訂標籤 API。如需更多詳細資訊，請參閱 AWS 受管政策：Amazon Rekognition 自訂標籤完全存取權限 。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

若要搭配最新版本的 Amazon Rekognition 自訂標籤使用 AWS SDK，您不再需要授予 Amazon Rekognition 自訂標籤許可，即可存取包含訓練和測試映像的 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：

- 中的使用者和群組 AWS IAM Identity Center：

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

  建立聯合身分的角色。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。請按照《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。此程式碼會列出您 AWS 帳戶中目前 AWS 區域中的專案。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。另請檢查 AWS 您用作 Amazon Rekognition 自訂標籤的區域並非所有 AWS 區域都可用。

若要呼叫 Amazon Rekognition 自訂標籤操作

1. 如果您尚未這麼做，請安裝並設定 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 加密的檔案 AWS Key Management Service。

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 加密的檔案 AWS Key Management Service

如果您使用 AWS Key Management Service (KMS) 來加密 Amazon Rekognition 自訂標籤清單檔案和映像檔案，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。

- kms:產生資料金鑰
- kms:解密

如需更多詳細資訊，請參閱 使用儲存在 AWS 金鑰管理服務 (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 自訂標籤會製作來源培訓和測試圖像的副本。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，則需要擁有該 KMS 金鑰的以下權限。

- kms:創建權限
- kms:描述金鑰

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 金鑰管理服務概念。您的來源圖像不會受到影響。

如需培訓模型的資訊，請參閱 培訓 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，資料集與專案的自動關聯就不會發生。

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。畫面將會顯示 Amazon Rekognition 自訂標籤的登入頁面。
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，請參閱從現有資料集複製內容。
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

本節提供工作流程的概觀，以訓練並使用 Amazon Rekognition 自訂標籤模型搭配 主控台和 AWS SDK。

> **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 主控台和 AWS SDK 為您的專案建立資料集。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。如需更多詳細資訊，請參閱 步驟 6：(可選) 將舊資料集與新專案建立關聯。

主題

- 決定模型類型
- 建立模型
- 改善模型
- 啟動模型
- 分析圖像
- 停止模型

決定模型類型

您首先決定要培訓的模型類型，這取決於您的業務目標。例如，您可以培訓模型在社交媒體貼文中尋找您的標誌、識別商店貨架上的產品或對裝配線上的機器零件進行分類。

Amazon Rekognition 自訂標籤可培訓以下類型的模型：

- 尋找物件、場景和概念
- 尋找物件位置
- 尋找品牌的位置

為了協助您決定要培訓的模型類型，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Amazon Rekognition 自訂標籤入門。

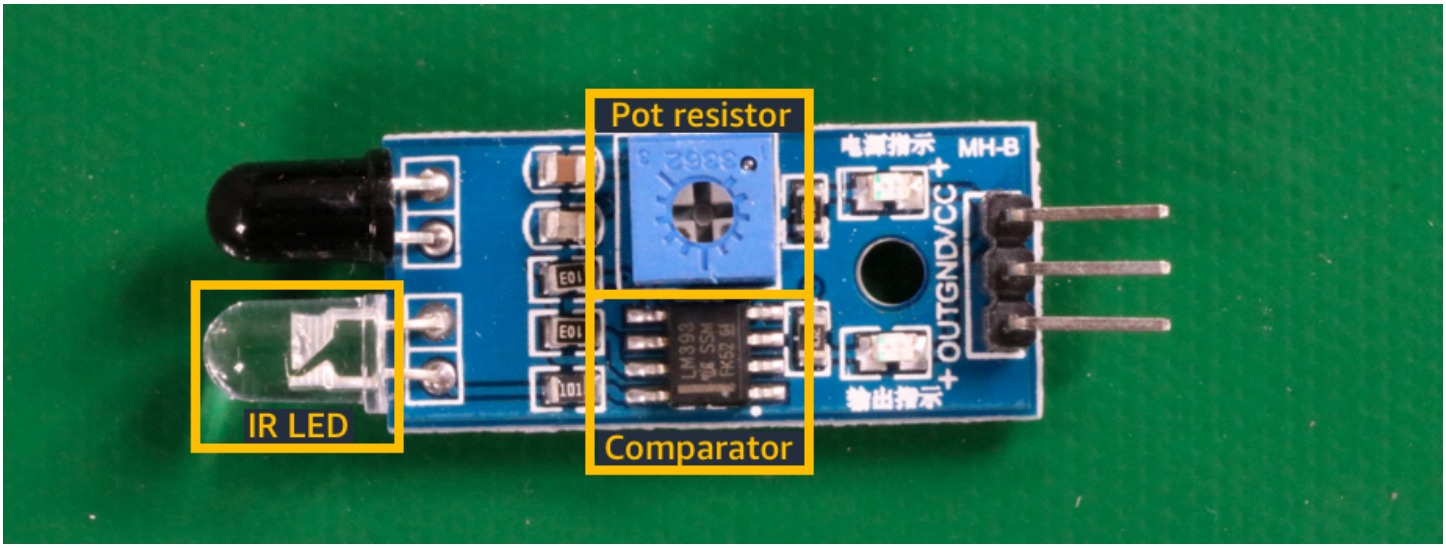
尋找物件、場景和概念

該模型預測與整個圖像相關聯的物件、場景和概念的分類。例如，您可以訓練一個模型來確定圖像是否包含 旅遊景點。如需範例專案，請參閱 Image classification。下列湖泊影像是您可以辨識物件、場景和概念的影像類型範例。

或者，您可以培訓一個將圖像分類為多個類別的模型。例如，上一張圖像可能具有 天空顏色、 反射 或 湖泊 等類別。如需範例專案，請參閱 多標籤影像分類。

尋找物件位置

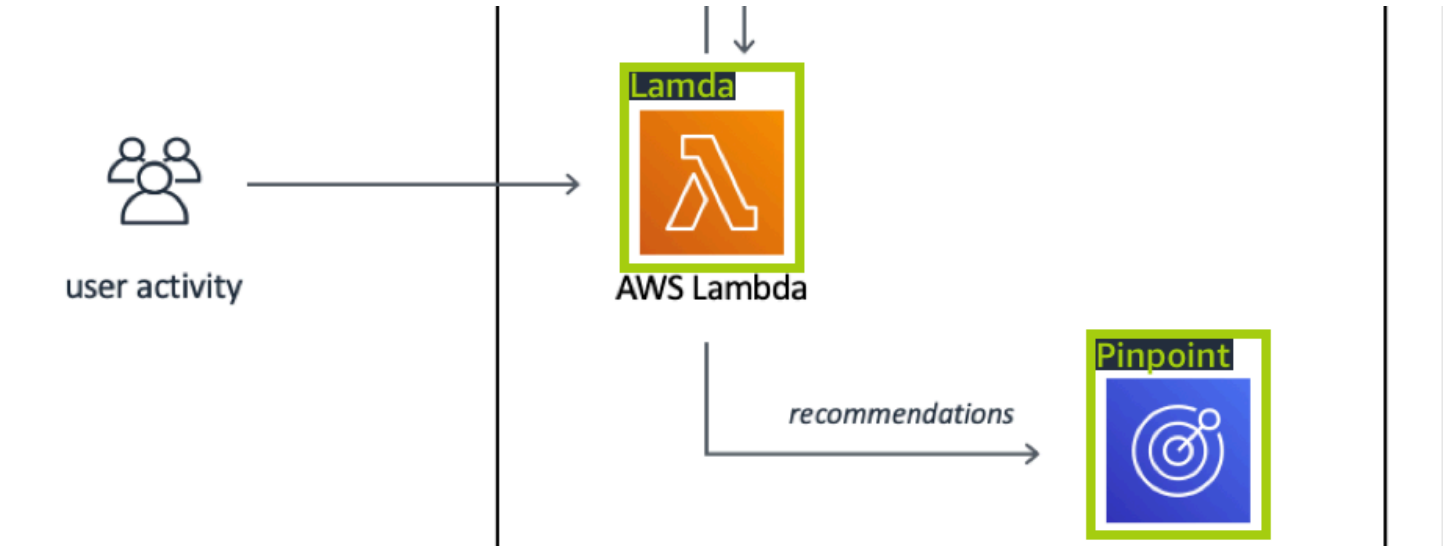
該模型會預測圖像上物件的位置。預測包括物件位置的邊界框資訊，以及用來識別邊界框內物件的標籤。例如，下圖顯示了電路板各個部分，例如 比較器 或 電位器 周圍的邊界框。



物件本地化 的範例專案展示了 Amazon Rekognition 自訂標籤如何使用標籤的邊界框來培訓尋找物件位置的模型。

尋找品牌的位置

Amazon Rekognition 自訂標籤可培訓在圖像上尋找品牌位置（例如商標）的模型。預測包括品牌位置的邊界框資訊，以及用來識別邊界框內物件的標籤。如需範例專案，請參閱 品牌偵測。下圖是模型可以偵測的一些品牌的範例。



建立模型

建立模型的步驟包括建立專案、建立培訓和測試資料集，以及培訓模型。

建立專案

Amazon Rekognition 自訂標籤專案是建立和管理模型所需的一組資源。專案會管理下列項目：

- 資料集 — 用來培訓模型的圖像和圖像標籤。專案備有培訓資料集和測試資料集。
- 模型 — 您培訓此軟體來尋找您的業務獨有的概念、場景和物件。您可以在一個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 建立專案 API 來建立專案。如需詳細資訊，請參閱建立專案。

建立培訓和測試資料集

資料集是描述這些影像的一組影像和標籤。在專案中，您可以建立培訓資料集和測試資料集，Amazon Rekognition 自訂標籤會使用它們來培訓和測試模型。

標籤可識別影像中物件周圍的物件、場景、概念或週框方塊。標籤會指定給整個圖像 (圖像層級)，或將標籤指定給圍繞圖像物件的邊界框。

> **Important**
>
> 如何標記資料集中的圖像，會決定 Amazon Rekognition 自訂標籤所建立的模型類型。例如，若要培訓尋找物件、場景和概念的模型，您可以為培訓和測試資料集中的圖像分配圖像層級標籤。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圖像的建議。如需詳細資訊，請參閱準備影像。

建立培訓和測試資料集 (主控台)

您可以使用單一資料集，或使用個別的培訓和測試資料集來啟動專案。如果您從單一資料集開始，Amazon Rekognition 自訂標籤會在培訓期間分割您的資料集，以便為您的專案建立培訓資料集 (80%) 和測試資料集 (20%)。如果您希望 Amazon Rekognition 自訂標籤決定要用於培訓和測試的圖像，請從單一資料集開始。為了完全控制培訓、測試和效能調整，我們建議您使用個別的培訓和測試資料集來啟動專案。

若要建立專案的資料集，請使用下列其中一種方式匯入圖像：

- 從本機電腦匯入圖像。
- 從 S3 儲存貯體匯入圖像。Amazon Rekognition 自訂標籤可以使用包含圖像的資料夾名稱來標記圖像。
- 匯入 Amazon SageMaker AI Ground Truth 資訊清單檔案。
- 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。

依據您匯入影像的位置而定，您的影像可能沒有標記。例如，從本機電腦匯入的影像即沒有標記。從 Amazon SageMaker AI 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。

若要使用主控台建立培訓和測試資料集，請參閱 建立包含影像的訓練和測試資料集。如需包含建立培訓和測試資料集的教學課程，請參閱 分類映像。

建立培訓和測試資料集 (SDK)

若要建立培訓和測試資料集，請使用 `CreateDataset` API。您可以使用 Amazon SageMaker 格式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。如需更多詳細資訊，請參閱 建立訓練和測試資料集 (SDK)。如有必要，您可以建立自己的清單檔案。如需詳細資訊，請參閱the section called “建立清單檔案”。

培訓模型

使用培訓資料集培訓您的模型。每次培訓模型時，都會建立新版本的模型。在培訓期間，Amazon Rekognition 自訂標籤會測試培訓模型的效能。您可以使用結果來評估和改善模型。培訓需要一段時間才能完成。您只需為成功的模型培訓付費。如需詳細資訊，請參閱培訓 Amazon Rekognition 自訂標籤模型。如果模型培訓失敗，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。

培訓模型（主控台）

若要使用主控台培訓模型，請參閱 培訓模型（主控台）。

培訓模型 (SDK)

您可以呼叫 建立專案版本 來培訓 Amazon Rekognition 自訂標籤模型。如需詳細資訊，請參閱培訓模型 (SDK)。

改善模型

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培訓過的模型。

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) 可讓您了解經過訓練的模型的效能，並決定是否準備好在生產中使用它。如需詳細資訊，請參閱用於評估模型的指標。

評估模型 (主控台)

如要檢視效能指標，請參閱 存取評估指標 (主控台)。

評估模型 (SDK)

要獲取效能指標，請呼叫 描述專案版本 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)。測試結果包括主控台中不可用的指標，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的整體效能的單一值。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，請參閱存取模型摘要檔案。
- 評估清單檔案快照位置 — 快照包含有關測試結果的詳細資料，包括信賴度分級和二進制分類測試的結果，例如誤報。`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，請參閱解譯評估資訊清單快照。

改善模型

如果需要改進，您可以新增更多培訓圖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饋，並使用它來改善模型。如需詳細資訊，請參閱使用模型意見回饋改善模型。

改善模型 (主控台)

若要新增影像至資料集，請參閱 將更多圖像新增至資料集。若要加入或變更標籤，請參閱 the section called “標記檔案”。

若要重新培訓模型，請參閱 培訓模型（主控台）。

改善模型 (SDK)

若要將圖像新增至資料集或變更圖像的標籤，請使用 UpdateDatasetEntries API。UpdateDatasetEntries 更新或將 JSON 文件添加至清單檔案。每個 JSON 文件都包含單一圖像的資訊，例如分配的標籤或邊界框資訊。如需詳細資訊，請參閱新增更多圖像 (SDK)。若要檢視資料集中的條目，請使用 ListDatasetEntries API。

若要重新培訓模型，請參閱 培訓模型 (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。您需要根據模型執行時間付費。如需詳細資訊，請參閱執行培訓過的模型。

啟動模型 (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)。

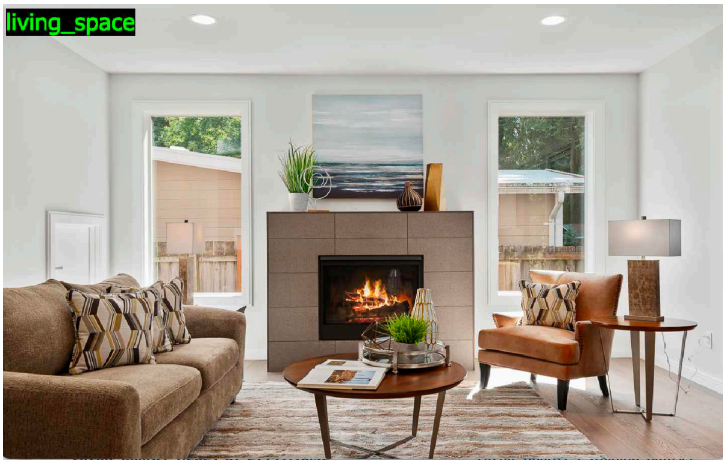
啟動模型

您可以呼叫 啟動模型版本 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)。

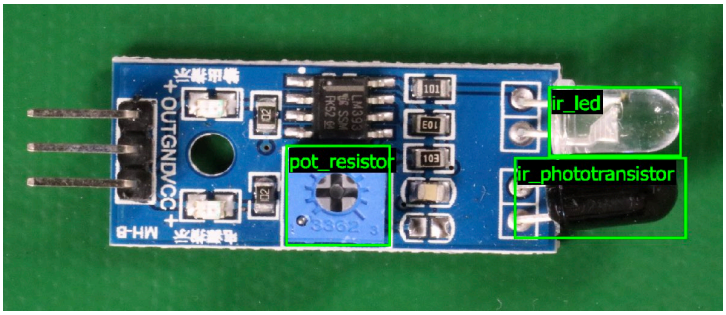
分析圖像

若要使用模型分析圖像，請使用 DetectCustomLabels API。您可以指定本機圖像或存放在 S3 儲存貯體中的圖像。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，則回應會包含在圖像中找到的圖像層級標籤清單。例如，下圖展示使用 Rooms 範例專案找到的圖像層級標籤。



如果模型找到物件位置，則回應會包含在圖像中找到的已標籤的邊界框清單。週框方塊代表物件在影像上的位置。您可以使用邊界框資訊，在物件周圍繪製邊界框。例如，以下圖像展示了使用 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

您需要根據模型的執行時間付費。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。

停止模型（主控台）

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)。

停止模型 (SDK)

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

Amazon Rekognition 自訂標籤入門

在開始這些入門指示之前，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，尋找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，或在印刷電路板上尋找電子零件的位置。

為了協助您開始使用，Amazon Rekognition 自訂標籤會包括教學課程影片和範例專案。

> **Note**
>
> 如需有關 Amazon Rekognition 自訂標籤支援 AWS 的區域和端點的資訊，請參閱 Rekognition 端點和配額。

教學課程影片

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

檢視教學課程影片

1. 登入 AWS 管理主控台 ，並在 https：//https://console.aws.amazon.com/rekognition/ 開啟 Amazon Rekognition 主控台。
2. 在左側視窗中，選擇 使用自訂標籤。畫面將會顯示 Amazon Rekognition 自訂標籤的登入頁面。如果沒有看到使用自訂標籤，請檢查您所使用的AWS 區域是否支援 Amazon Rekognition 自訂標籤。
3. 從導覽窗格選擇開始使用。
4. 在什麼是 Amazon Rekognition 自訂標籤？中，選擇影片，以觀看概觀影片。
5. 在導覽窗格中，選擇教學課程。
6. 在教學課程頁面上，選擇您要觀看的教學課程影片。

範例專案

Amazon Rekognition 自訂標籤提供下列範例專案。

Image classification

影像分類專案 (Rooms) 會訓練模型，使模型可在影像中尋找一個或多個家庭位置，例如後院、廚房和庭院。訓練和測試影像代表單一位置。每個影像都會以單一影像層級標籤標記，例如廚房、庭院或 living_space。針對分析的影像，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，模型可能會在下列影像中尋找標籤 living_space。如需詳細資訊，請參閱尋找物件、場景和概念。

多標籤影像分類

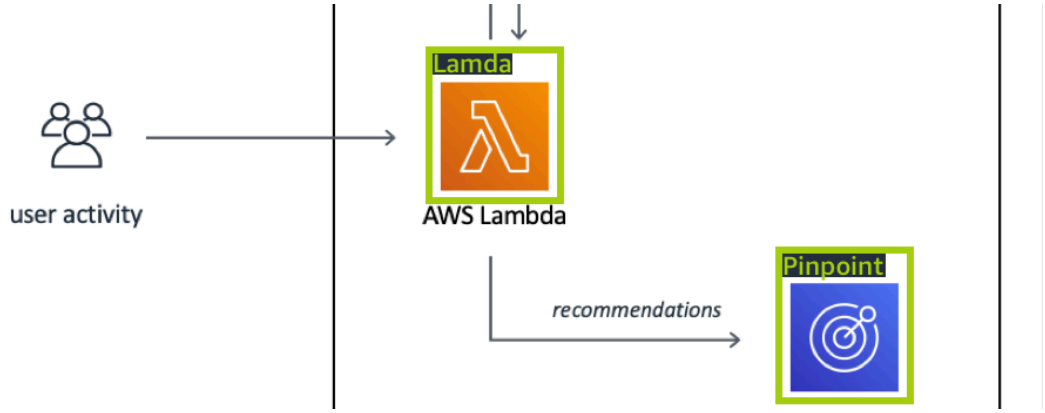
多標籤影像分類專案 (Flowers) 會訓練模型，將花卉的影像分類為三個概念 (花卉類型、有葉狀態和生長階段)。

訓練和測試影像具有適用於每個概念的影像層級標籤，例如適用於花卉類型的山茶花、適用於帶葉花卉的 with_leaves，以及適用於成熟花卉的 fully_growth。

針對分析的影像，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，模型會針對下列影像傳回標籤 mediterranean_spurge 和 with_leaves。如需詳細資訊，請參閱尋找物件、場景和概念。

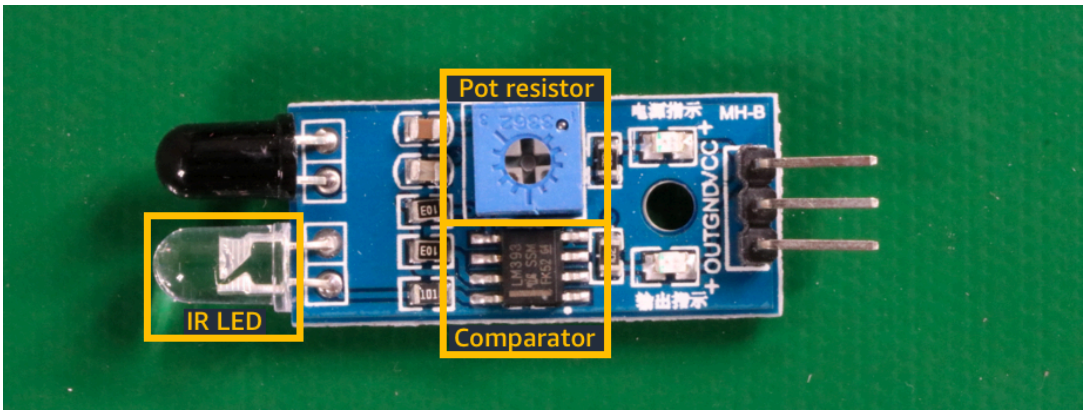
品牌偵測

品牌偵測專案 (Logos) 會訓練模型，該模型會尋找特定 AWS 標誌的位置，例如 Amazon Textract 和 AWS lambda。訓練影像僅屬於標誌，並具有單一影像層級標籤，例如 lambda 或 textract。它也可能使用具有品牌位置週框方塊的訓練影像來訓練品牌偵測模型。測試影像已標記週框方塊，代表標誌在自然位置 (例如架構圖) 的位置。訓練過的模型會尋找標誌，並針對找到的每個標誌傳回標記的週框方塊。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 (Circuit boards) 會訓練模型，該模型可在印刷電路板上尋找零件的位置，例如比較器或紅外線發光二極體。訓練和測試影像包括圍繞電路板零件的週框方塊，以及用於識別週框方塊內零件的標籤。在下列範例影像中，標籤名稱為 ir_phototransistor、ir_led、pot_resistor 和 comparator。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標記的邊框。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」指示會說明如何使用 Amazon Rekognition 自訂標籤為您建立的範例專案來訓練模型。它還會向您顯示如何啟動模型並將其用於分析影像。

建立範例專案

若要開始使用，請決定要使用哪個專案。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，以及標記所有影像的測試資料集。您在訓練模型之前不需要進行任何變更。範例專案會顯示 Amazon Rekognition 自訂標籤使用標籤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。
- 週框方塊 — 標籤可識別週框方塊的內容。週框方塊是一組圍繞影像中物件的影像座標。

稍後，當您使用自己的影像建立專案時，您必須建立訓練和測試資料集，也必須標記您的影像。如需詳細資訊，請參閱決定模型類型。

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：培訓您的模型。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，因此您無需在執行模型之前評估模型。如需詳細資訊，請參閱改善訓練過的 Amazon Rekognition 自訂標籤模型。

使用模型

接下來，啟動模型。如需詳細資訊，請參閱步驟 3：啟動模型。

開始執行模型後，您即可將其用於分析新影像。如需詳細資訊，請參閱步驟 4：使用模型分析影像。

您需要根據模型執行時間付費。使用完範例模型後，應停止模型。如需更多詳細資訊，請參閱 步驟 5：停止模型。

後續步驟

您可以在準備好時建立自己的專案。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，您會使用「選擇範例專案」。Amazon Rekognition 自訂標籤接著即會為您建立專案和資料集。專案會管理用於訓練模型的檔案。如需詳細資訊，請參閱管理 Amazon Rekognition 自訂標

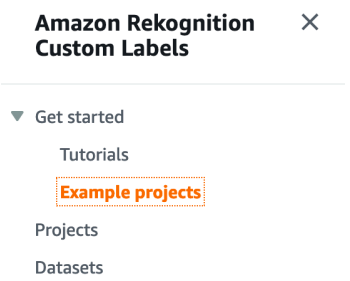
籤專案。資料集包含用於訓練和測試模型的影像、指定的標籤和週框方塊。如需詳細資訊，請參閱the section called “管理資料集”。

如需有關範例專案的詳細資料，請參閱 範例專案。

選擇範例專案

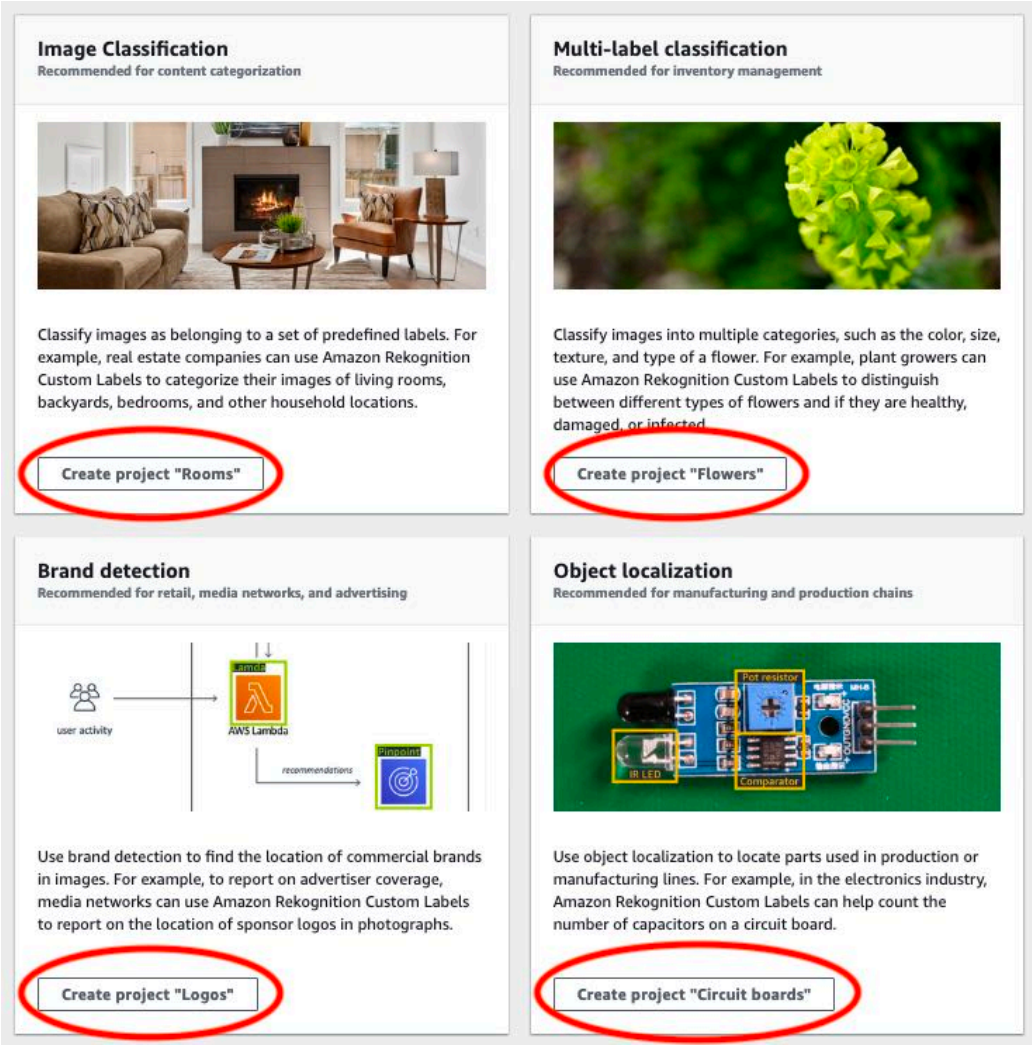
1. 登入 AWS 管理主控台 ，並在 https：//https://console.aws.amazon.com/rekognition/ 開啟 Amazon Rekognition 主控台。
2. 在左側視窗中，選擇 使用自訂標籤。畫面將會顯示 Amazon Rekognition 自訂標籤的登入頁面。如果沒有看到使用自訂標籤，請檢查您所使用的AWS 區域是否支援 Amazon Rekognition 自訂標籤。
3. 選擇開始使用。

   Amazon Rekognition 自訂標籤區段顯示入門、反白顯示「範例專案」的教學課程、專案和資料集。

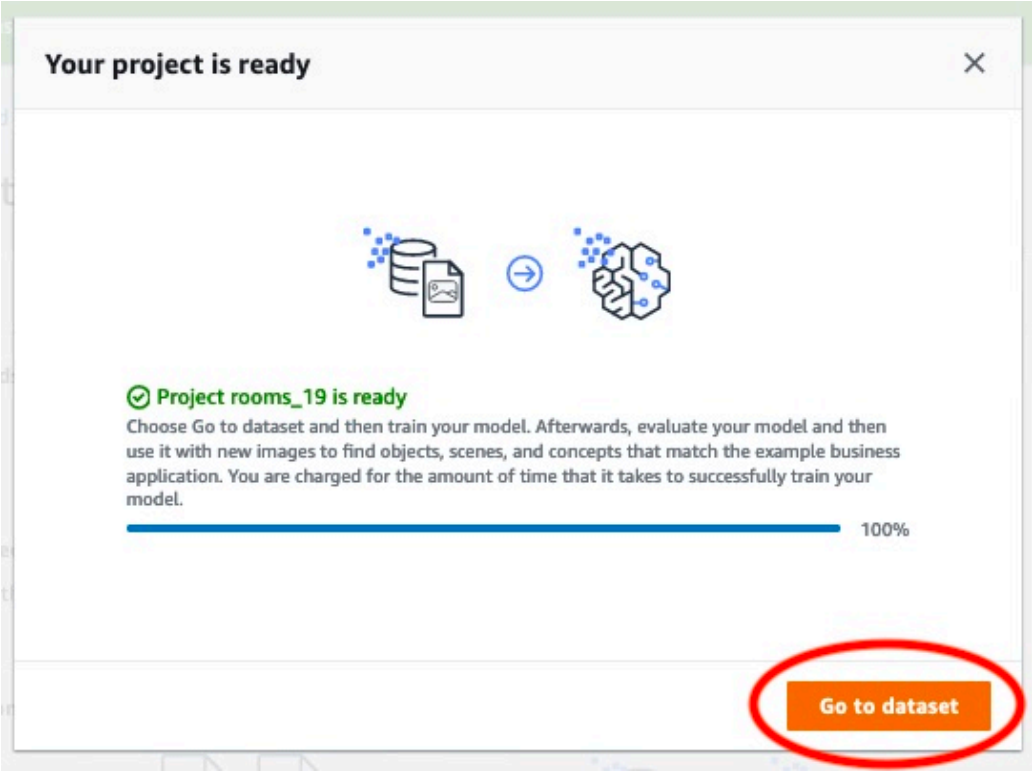


4. 在探索範例專案中，選擇試用範例專案。
5. 決定您要使用的專案，然後在範例區段中選擇建立專案「####」。Amazon Rekognition 自訂標籤接著即會為您建立範例專案。

> **Note**
>
> 如果這是您第一次在目前 AWS 區域中開啟主控台，則會顯示第一次設定對話方塊。請執行下列操作：
>
> 1. 請記下所顯示的 Amazon S3 儲存貯體名稱。
> 2. 選擇繼續，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)。以下主控台的影像顯示具有「建立專案」按鈕的範例，用於影像分類 （房間）、多標籤分類 （花朵）、品牌偵測 （標誌） 和物件本地化 （電路板）。



6. 專案準備就緒後，請選擇前往資料集。下圖顯示專案就緒時專案面板的外觀。

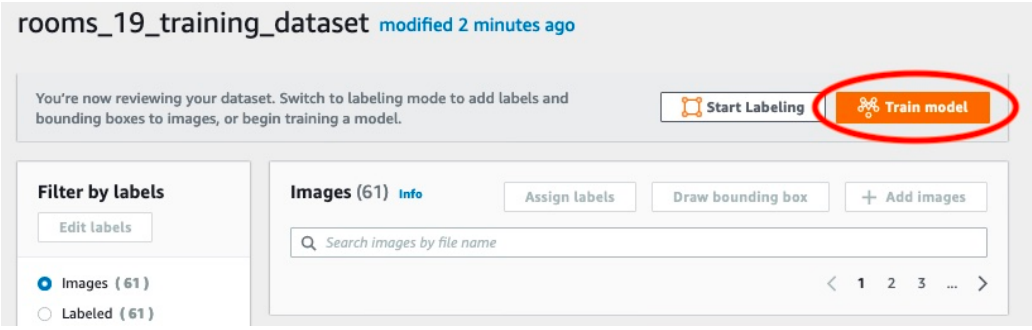


步驟 2：培訓您的模型

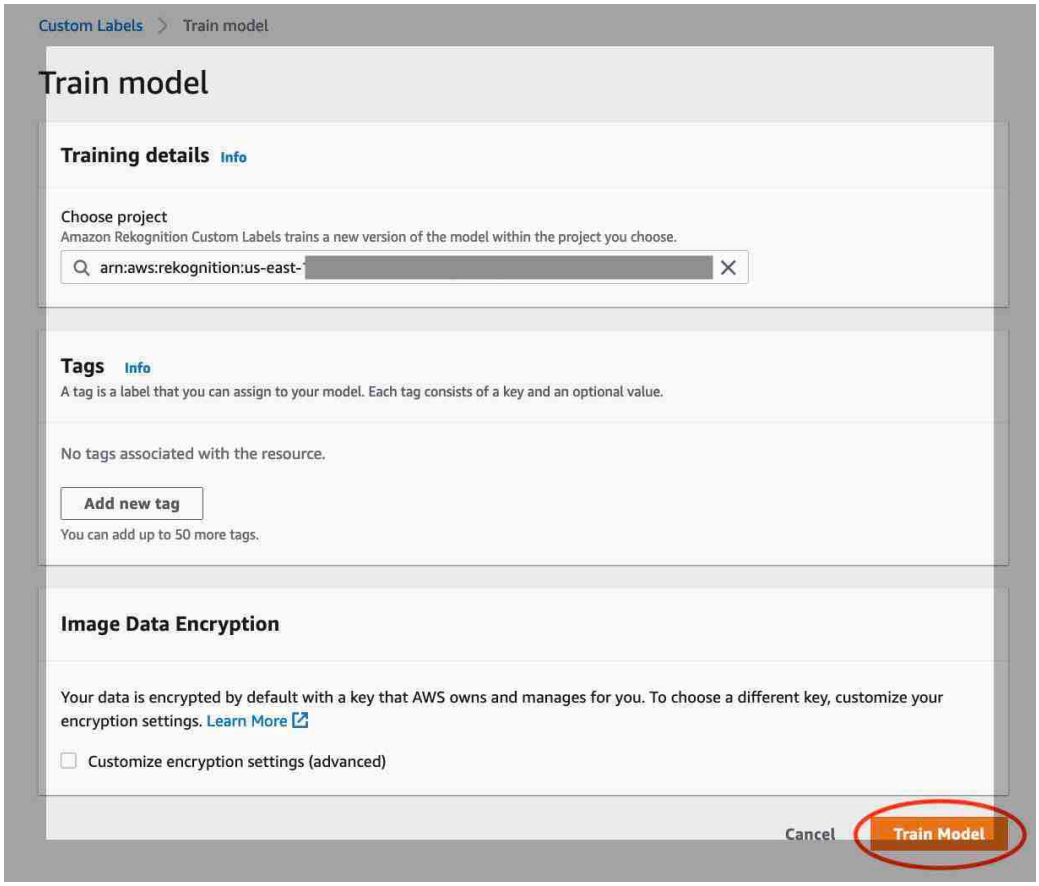
在此步驟中，您會訓練模型。您的訓練和測試資料集會自動設定。訓練成功完成後，您可以查看整體評估結果，以及個別測試影像的評估結果。如需詳細資訊，請參閱培訓 Amazon Rekognition 自訂標籤模型。

訓練您的模型

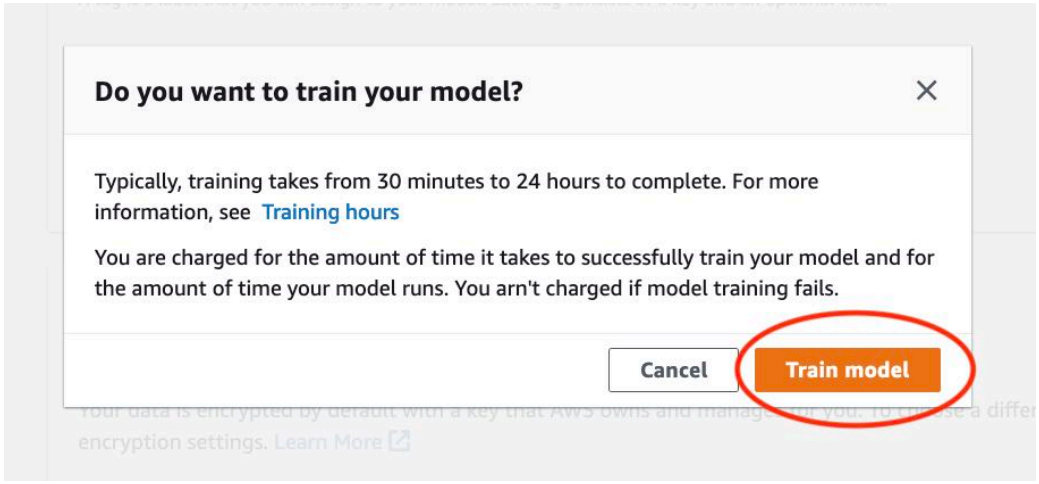
1. 在資料集頁面上，選擇訓練模型。下圖顯示具有訓練模型按鈕的 主控台。



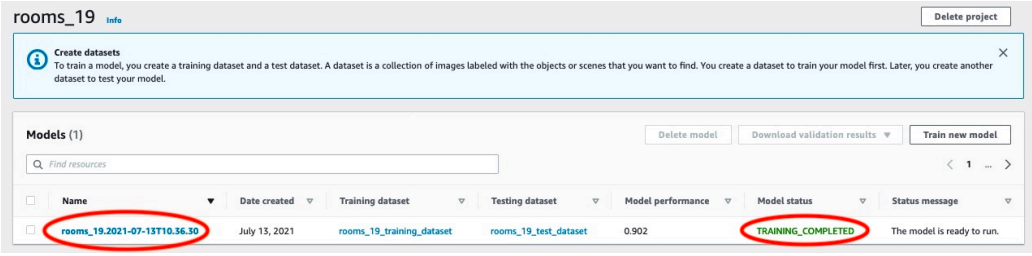
2. 在訓練模型 的頁面上，選擇訓練模型。下圖顯示訓練模型按鈕，請注意您專案的 Amazon Resource Name (ARN) 位於選擇專案編輯方塊中。



3. 在您想要訓練模型嗎？對話方塊中，如下圖所示，選擇訓練模型。

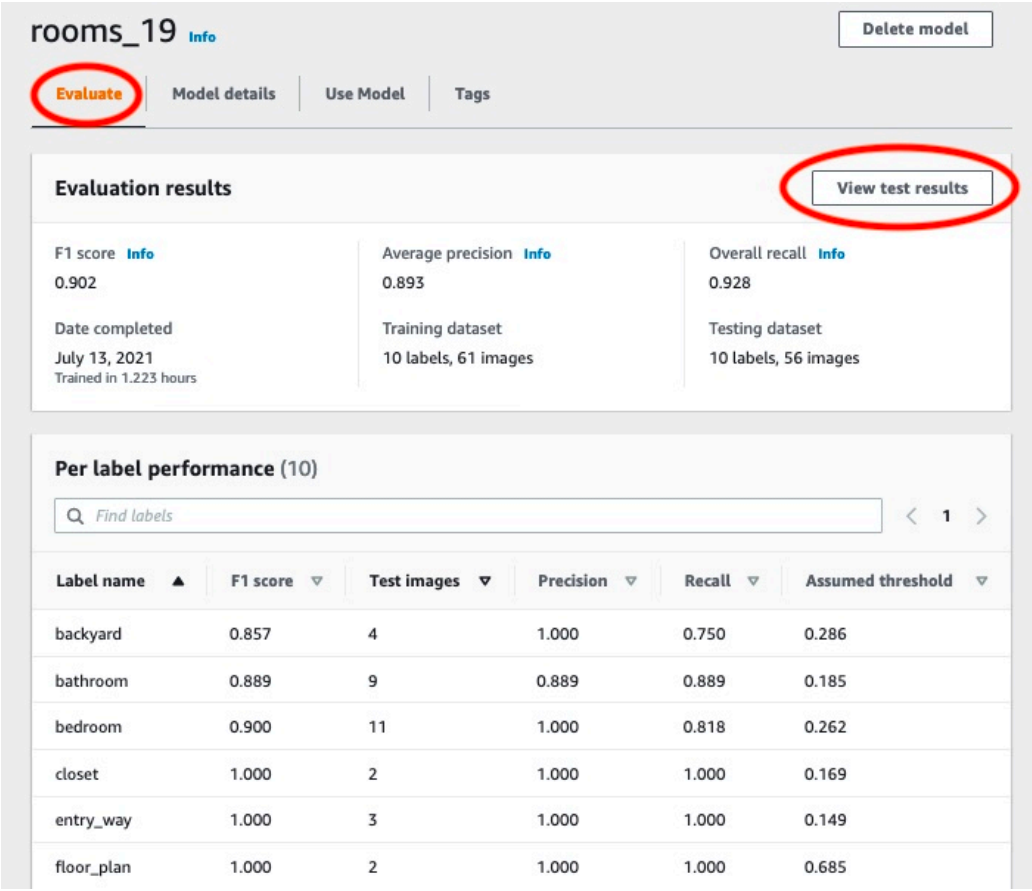


4. 訓練完成後，請選擇模型名稱。當模型狀態為 TRAINING_COMPLETED 時，訓練即完成，如下列主控台螢幕擷取畫面所示。

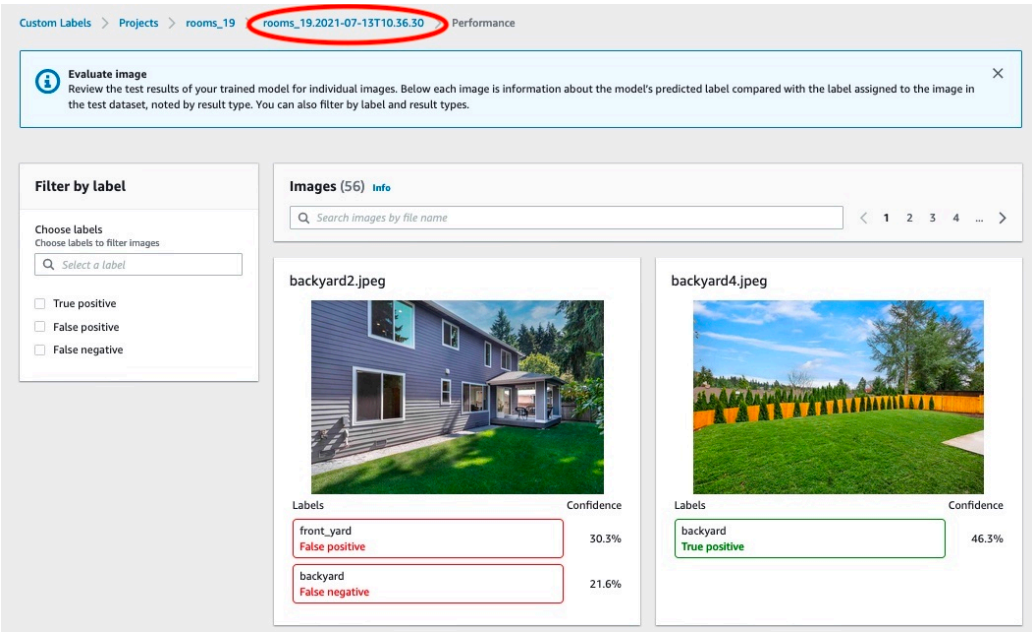


5. 選擇評估按鈕以查看評估結果。如需評估模型的資訊，請參閱 改善訓練過的 Amazon Rekognition 自訂標籤模型。

6. 選擇檢視測試結果，以查看個別測試影像的結果。如下列螢幕擷取畫面所示，評估儀表板會顯示指標，例如 F1 分數、精確度和每個標籤的召回，以及測試影像的數量。也會顯示整體指標，例如平均值、精確度和召回率。



7. 查看測試結果後，選擇模型名稱以返回模型頁面。下列效能儀表板的螢幕擷取畫面，您可以在其中按一下以返回模型頁面。



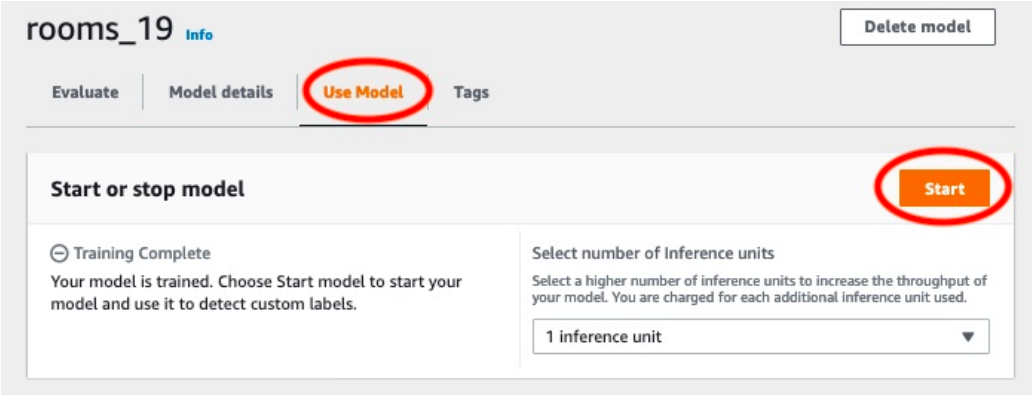
步驟 3：啟動模型

在此步驟中，您可以啟動模型。模型啟動後，您即可將其用於分析影像。

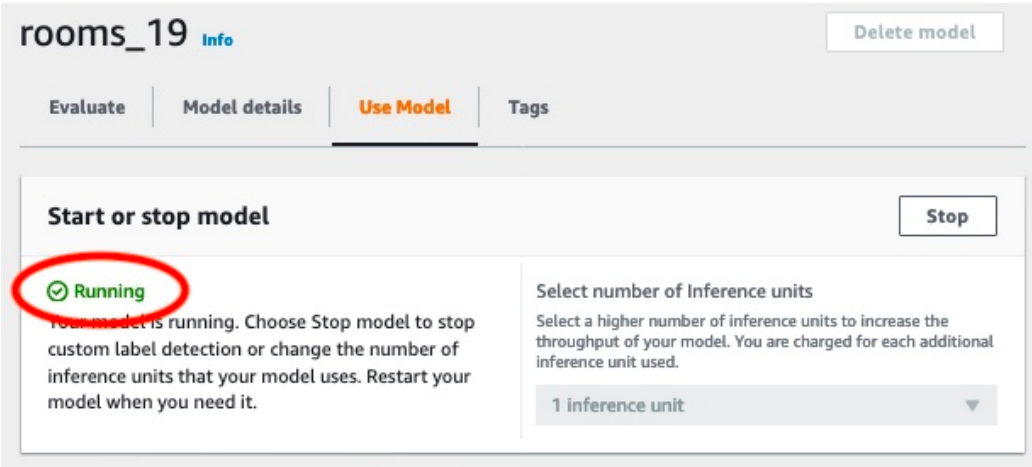
您需要根據模型執行時間付費。如果您不需要分析影像，請停止您的模型。您可以在稍後時間重新啟動您的模型。如需詳細資訊，請參閱執行培訓過的 Amazon Rekognition 自訂標籤模型。

啟動模型

1. 選擇模型頁面上的使用模型索引標籤。
2. 在啟動或停止模型區段中，執行以下操作：
   a. 選擇 啟動。
   b. 在啟動模型的對話框中，選擇啟動。下圖顯示模型控制面板中的開始按鈕。



3. 等待模型執行。下列螢幕擷取畫面顯示執行模型時的主控台，其中開始或停止模型區段中的狀態正在執行。



4. 使用模型來分類影像。如需詳細資訊，請參閱步驟 4：使用模型分析影像。

步驟 4：使用模型分析影像

呼叫 DetectCustomLabels API 以分析影像。在此步驟中，您會使用 `detect-custom-labels` AWS Command Line Interface (AWS CLI) 命令來分析範例映像。您可以從 Amazon Rekognition 自訂標籤主控台取得 AWS CLI 命令。主控台會將 AWS CLI 命令設定為使用您的模型。您只需要提供儲存在 Amazon S3 儲存貯體中的影像。本主題會提供可用於每個範例專案的影像。

> **Note**
>
> 主控台還會提供 Python 範例程式碼。

來自 `detect-custom-labels` 的輸出包括在影像中找到的標籤清單、邊界框 (如果模型尋找物體位置)，以及模型對預測準確度的信賴度。

如需詳細資訊，請參閱使用經過培訓的模型分析圖像。

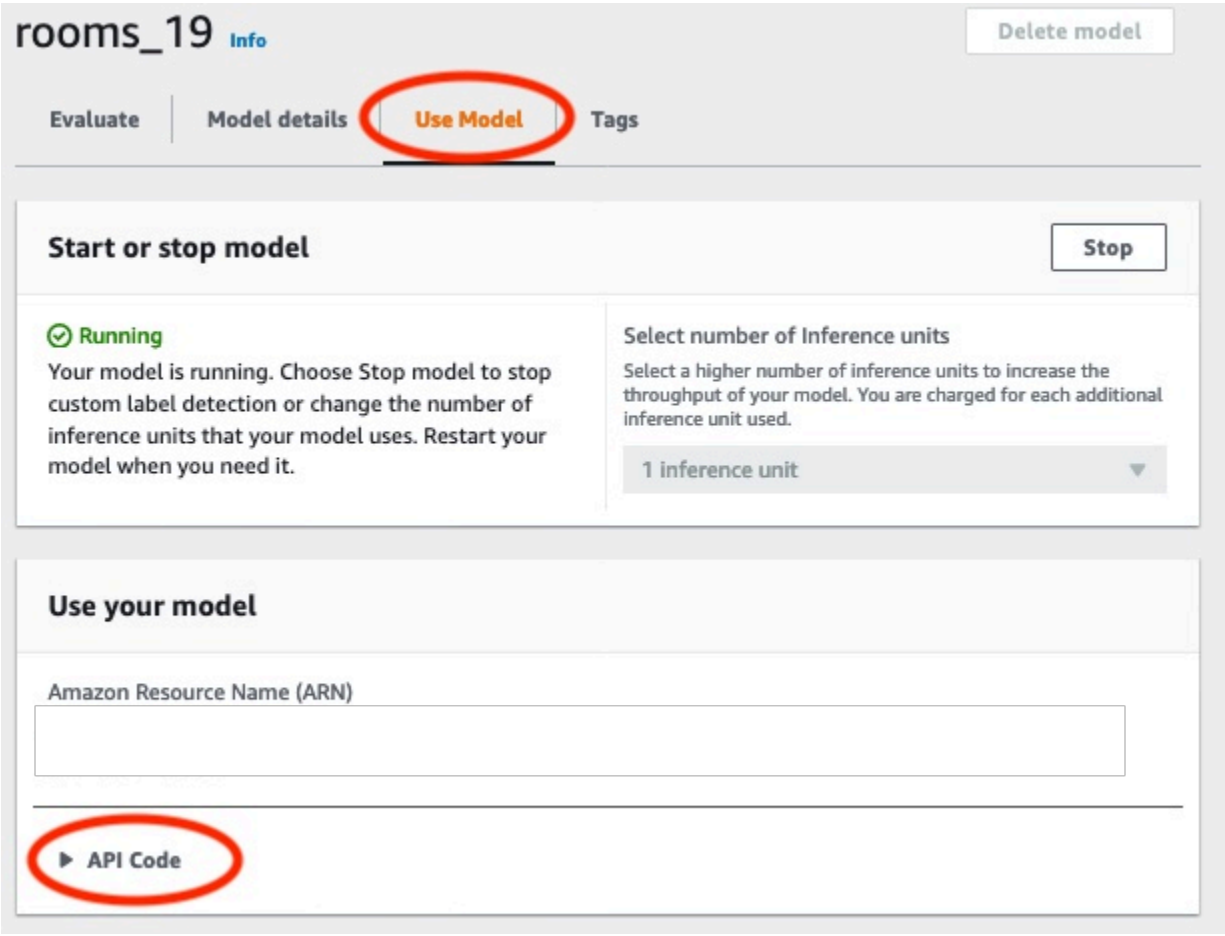
分析影像 (主控台)

1. <textobject><phrase>顯示為執行中的模型狀態，使用停止按鈕停止執行中的模型。</phrase></textobject>

   如果您尚未設定 ，請設定 AWS CLI。如需說明，請參閱the section called “步驟 4：設定 AWS CLI 和 AWS SDKs”。

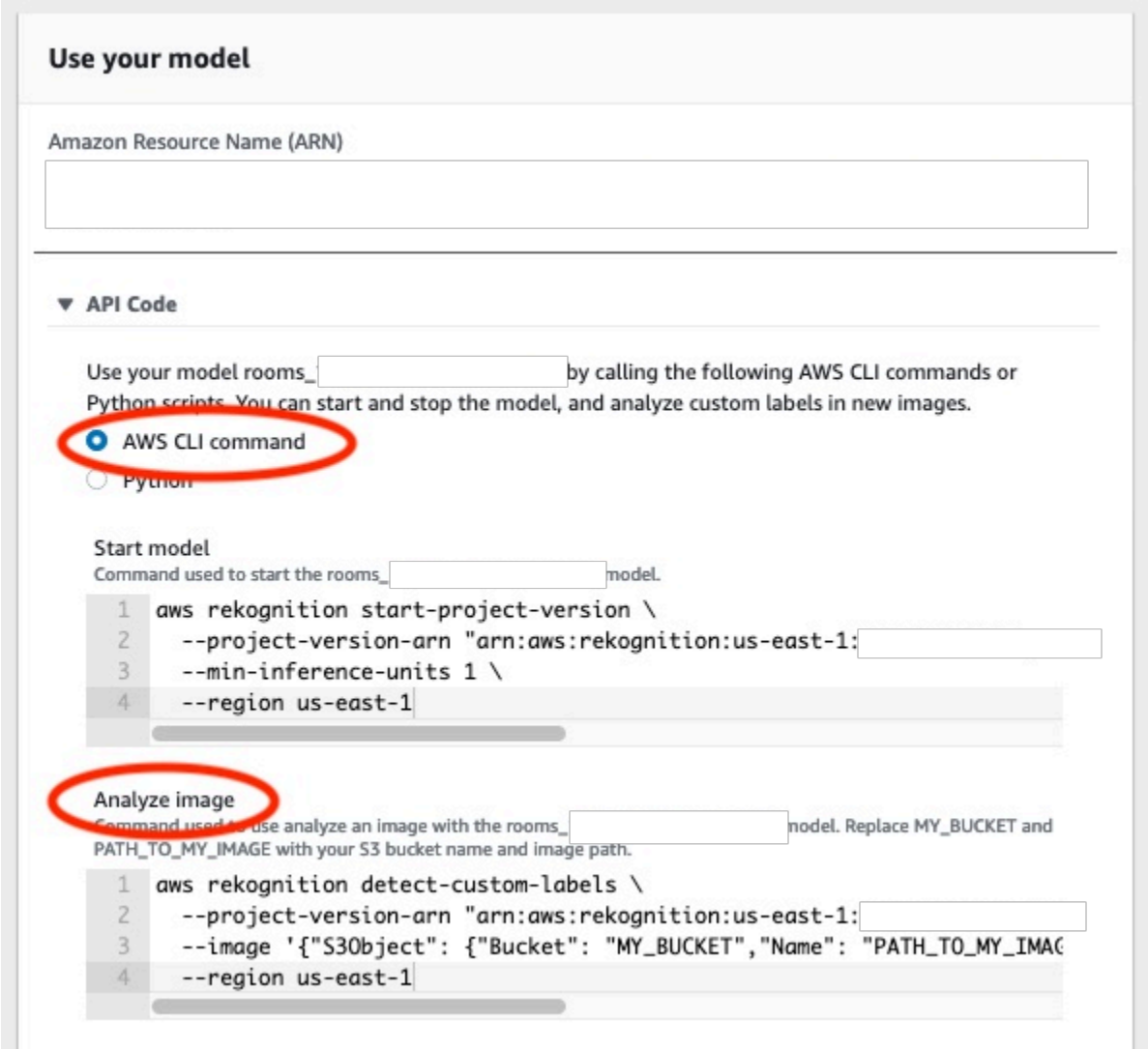
2. 如果您尚未執行，請開始執行模型。如需詳細資訊，請參閱步驟 3：啟動模型。

3. 選擇使用模型索引標籤，然後選擇 API 程式碼。以下顯示的模型狀態面板顯示模型為執行中，並顯示停止按鈕來停止執行中的模型，以及顯示 API 的選項。



4. 選擇 AWS CLI 命令。

5. 在分析映像區段中，複製呼叫 的 AWS CLI 命令`detect-custom-labels`。下圖的 Rekognition 主控台顯示「分析影像」區段，其中包含使用機器學習模型偵測影像上自訂標籤的 AWS CLI 命令，以及啟動模型並提供影像詳細資訊的說明。



6. 將範例影像上傳至 Amazon S3 儲存貯體。如需說明，請參閱取得範例影像。

7. 在命令提示字元中，輸入您在上一個步驟中複製的 AWS CLI 命令。輸出應該如以下範例所示。

   `--project-version-arn` 的值應該是模型的 Amazon Resource Name (ARN)。`--region` 的值應該是您在其中建立模型的 AWS 區域。

   將 `MY_BUCKET` 和 `PATH_TO_MY_IMAGE` 變更為您在上一步驟中所使用的 Amazon S3 儲存貯體和影像。

   如果您是使用 自訂標籤存取 設定檔來取得憑證，請新增 `--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
```

如果模型尋找物件、場景和概念，則來自 AWS CLI 命令的 JSON 輸出看起來應類似下列內容。Name 是模型找到的影像層級標籤名稱。Confidence (0-100) 是模型在預測準確性方面的可信度。

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

如果模型尋找物件位置或尋找品牌，則會傳回標記的週框方塊。BoundingBox 包含物件周圍方塊的位置。Name 是模型在週框方塊中找到的物件。Confidence 是模型對於週框方塊包含物件的可信度。

```
{
    "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。如果不再使用，請停止模型。如需詳細資訊，請參閱步驟 5：停止模型。

取得範例影像

您可以使用下列影像進行 DetectCustomLabels 操作。每個專案都有一個影像。若要使用影像，您可以將其上傳到 S3 儲存貯體。

使用範例影像

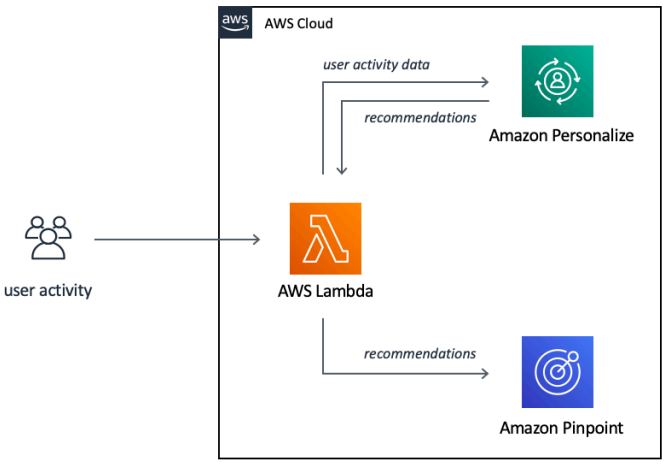
1. 在以下與您正在使用的範例專案相符的影像上按一下右鍵。然後選擇儲存影像，將影像儲存到電腦。依您使用的瀏覽器而定，功能表選項可能會有所不同。
2. 將映像上傳至您 AWS 帳戶擁有的 Amazon S3 儲存貯體，且位於您使用 Amazon Rekognition 自訂標籤的相同 AWS 區域。

   如需指示說明，請參閱 Amazon 簡單儲存服務使用者指南中的將物件上傳至 Amazon S3。

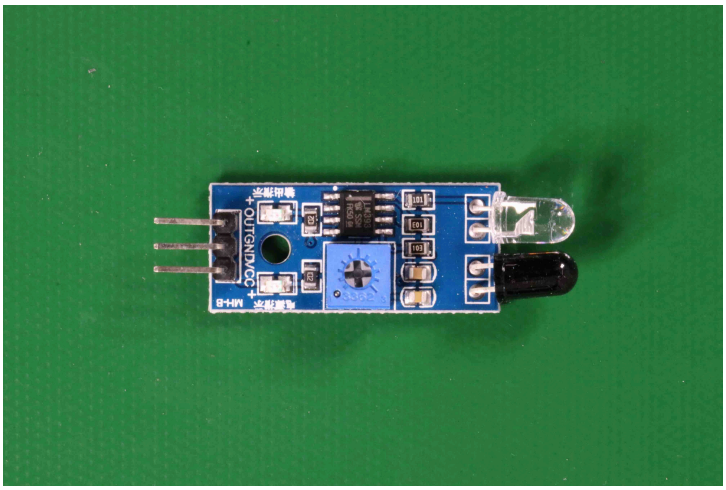
Image classification

多標籤分類

品牌偵測



物件本地化

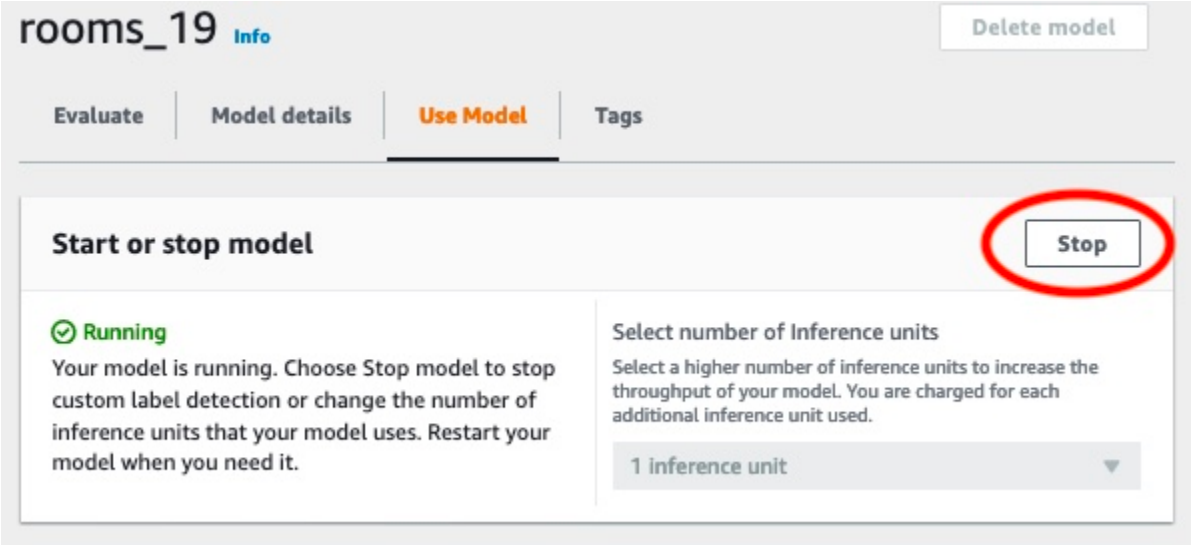


步驟 5：停止模型

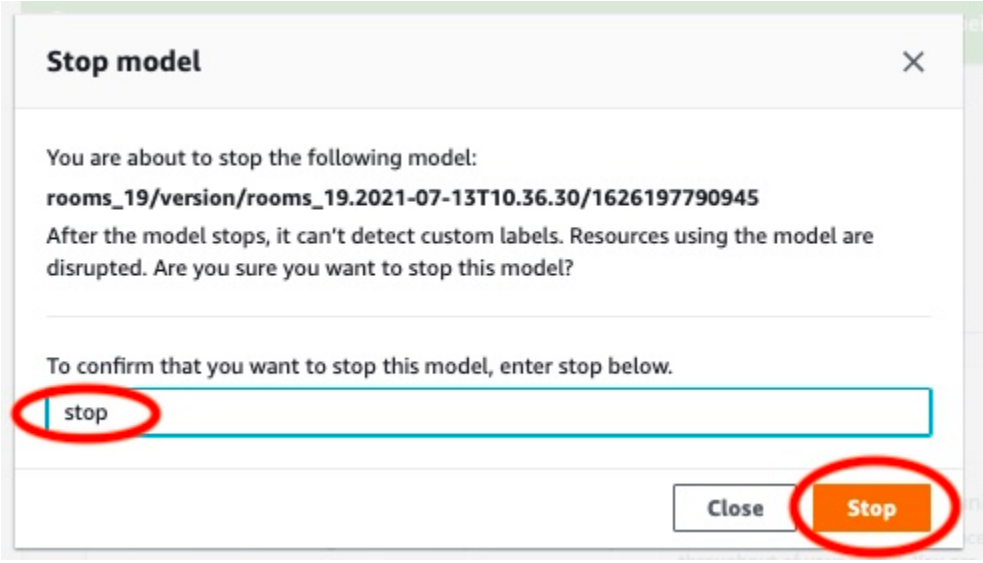
在此步驟中，您將停止執行模型。您需要根據模型執行時間付費。如果您已使用完成，請停止模型。

停止模型

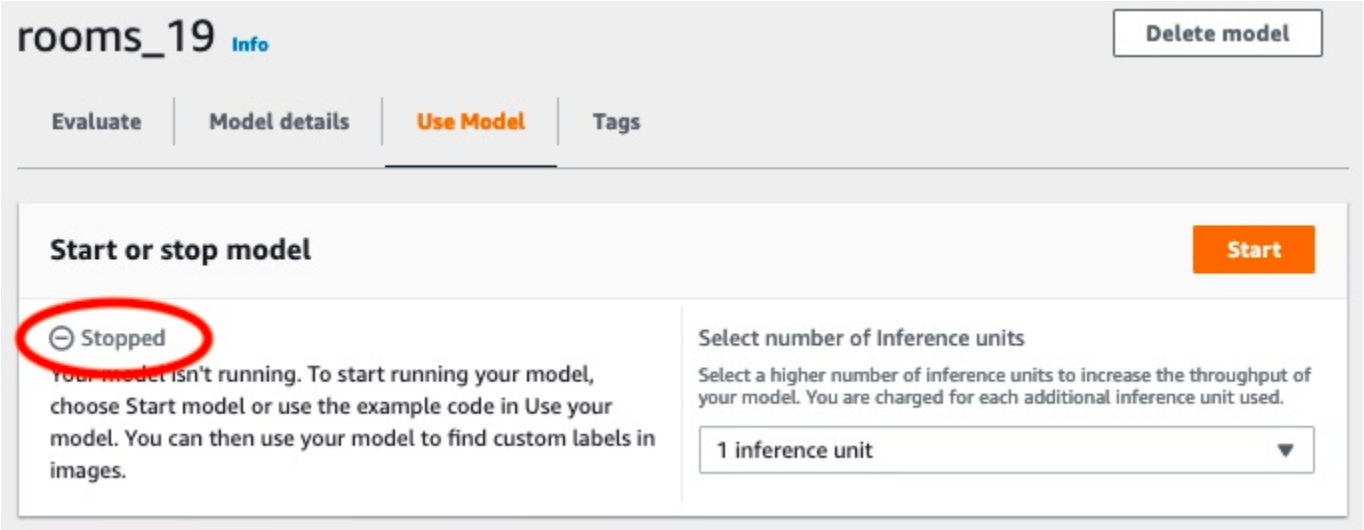
1. 在啟動或停止模型區段中，選擇停止。



2. 在停止模型對話框中，輸入 停止，以確認您要停止模型。



3. 選擇停止，以停止模型。當 啟動或停止模型 的區段中的狀態顯示為 已停止 時，模型即已停止。在下列螢幕擷取畫面中，使用者介面區段可以選擇啟動或停止機器學習模型。模型的狀態會顯示為「已停止」，並顯示「開始」按鈕來啟動模型，並顯示下拉式清單來選取推論單位的數量。



步驟 6：後續步驟

試用完範例專案後，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]。

使用下表中的標記資訊來訓練與範例專案類似的模型。

範例	訓練影像	測試影像
影像分類 (Rooms)	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 (Flowers)	每個影像多個影像層級標籤	每個影像多個影像層級標籤
品牌檢測 (Logos)	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
影像本地化 (Circuit boards)	標記的週框方塊	標記的週框方塊

[分類映像] 會說明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 自訂標籤模型]。

分類映像

本教學課程說明如何為模型建立專案和資料集，以分類影像中的物體、場景和概念。模型會對整個影像進行分類。例如，透過跟隨此教學課程，您可以培訓模型以辨識家居位置，例如客廳或廚房。本教學課程還會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，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的影像來建立培訓和測試資料集。稍後，您可以為培訓和測試資料集中的影像指派影像層級標籤。

您建立的模型將影像分類為屬於您指派給培訓資料集影像的影像層級標籤合集。例如，培訓資料集中的影像層級標籤集為 kitchen、 living_room、 patio 和 backyard，則模型可能會在單一影像中找到所有這些影像層級標籤。

> **Note**
>
> 您可以為不同目的建立模型，例如尋找影像上物體的位置。如需詳細資訊，請參閱決定模型類型。

步驟 1：收集您的影像

您需要兩組影像。一組可新增至培訓資料集。另一組則添加至您的測試資料集。影像應代表您希望模型分類的物體、場景和概念。影像必須是 PNG 或 JPEG 的格式。如需詳細資訊，請參閱準備影像。

您應該至少有 10 個用於培訓資料集的影像和 10 個用於測試資料集的影像。

如果您還沒有影像，請使用 Rooms 範例分類專案中的影像。建立專案後，培訓和測試影像會位於以下 Amazon S3 儲存貯體的位置：

- 培訓圖片 — 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 是您使用 Amazon Rekognition 自訂標籤主控台 AWS 的區域。 numbers 是主控台指派給儲存貯體名稱的值。 Version number是範例專案的版本編號，從 1 開始。

以下步驟將 Rooms 專案中的影像儲存到電腦上名為 `training` 和 `test` 的本機資料夾。

下載 Rooms 範例專案影像文件

1. 建立 Rooms 專案。如需詳細資訊，請參閱步驟 1：選擇範例專案。
2. 開啟命令提示字元並輸入以下命令來下載培訓影像。

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

3. 在命令提示字元處，輸入以下命令以下載測試影像。

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

4. 將培訓資料夾中的兩個影像移至您選擇的單獨資料夾。您將使用這些影像來嘗試在 步驟 9：使用模型分析影像 中培訓的模型。

步驟 2：決定課程

列出您希望模型尋找的課堂清單。例如，您正在培訓模型來識別房屋中的房間，則可以將以下影像分類為 `living_room`。

每個類別對應到一個影像層級標籤。稍後，您可以為培訓和測試資料集中的影像指派影像層級標籤。

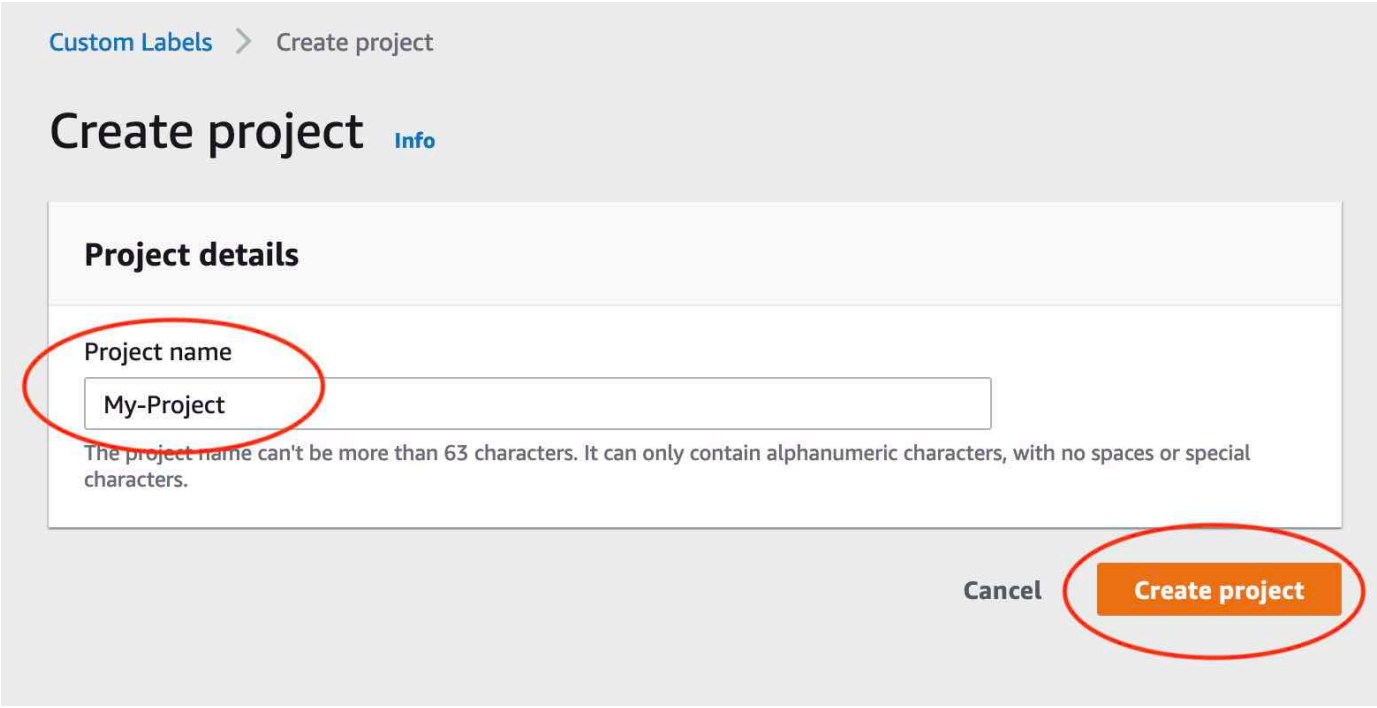
如果您使用 Rooms 範例專案中的影像，則影像層級標籤將會是 後院、浴室、睡房、衣櫃、入口通道、樓層平面圖、前院、廚房、生活空間 和 露臺。

步驟 3：建立專案

要管理您的資料集和模型，您需要建立一個專案。每個專案都應該解決一個使用案例，例如識別房屋中的房間。

建立專案 (主控台)

1. 如果您尚未這麼做，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 管理主控台 ，並在 https：//https://console.aws.amazon.com/rekognition/ 開啟 Amazon Rekognition 主控台。
3. 在左側視窗中，選擇 使用自訂標籤。畫面將會顯示 Amazon Rekognition 自訂標籤的登入頁面。
4. 在 Amazon Rekognition 自訂標籤登入頁面，選擇 開始使用
5. 在左側導覽視窗中，選擇 專案。
6. 在專案頁面上，選擇 建立專案。
7. 在 專案名稱 中，輸入您的專案名稱。
8. 選擇 建立專案 以建立您的專案。



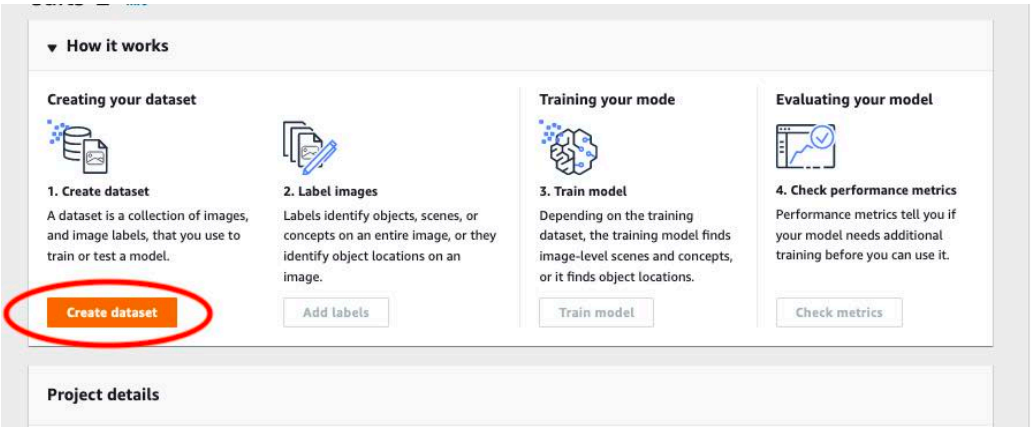
步驟 4：建立培訓和測試資料集

在此步驟中，您透過從本機電腦上傳影像來建立培訓資料集和測試資料集。您一次最多可以上傳 30 張圖片。如果您有大量影像需要上傳，請考慮透過從 Amazon S3 儲存貯體匯入影像來建立資料集。如需更多詳細資訊，請參閱 從 Amazon S3 儲存貯體匯入映像。

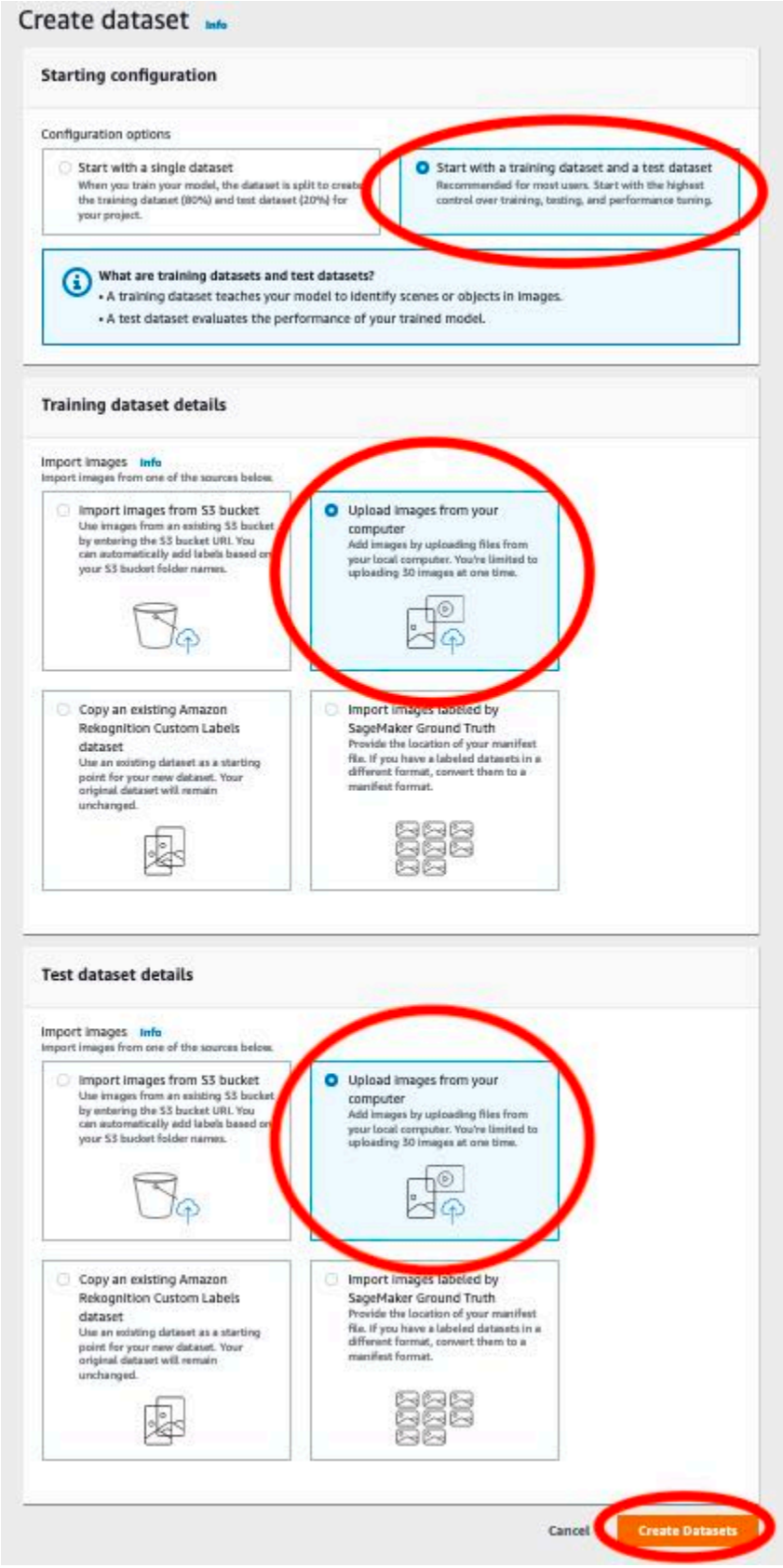
如需更多有關資料集的詳細資訊，請參閱 管理資料集。

使用本機電腦 (主控台) 上的影像建立資料集

1. 在專案的詳細頁面上，選擇 建立資料集。

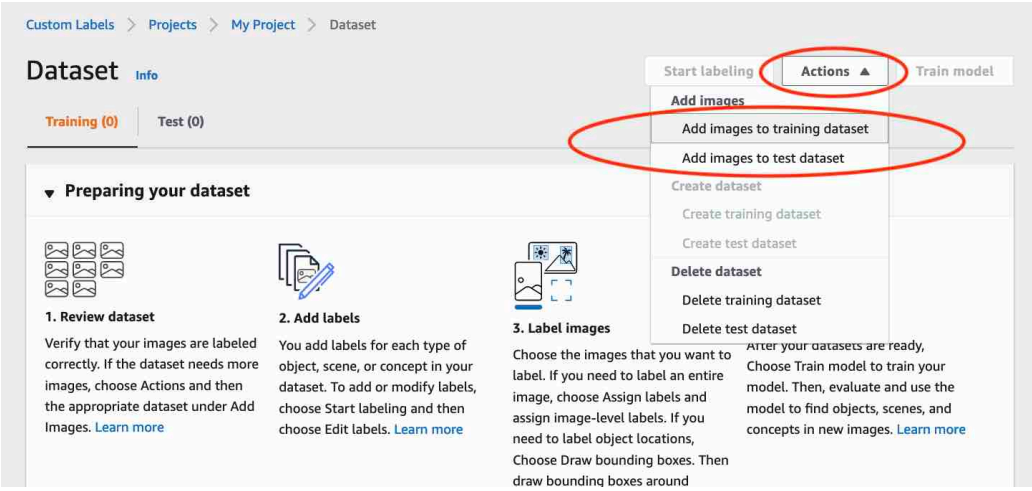


2. 在 開始配置 的區段中，選擇 從培訓資料集和測試資料集開始。
3. 在 培訓資料集詳細資料 的區段中，選擇 從電腦上傳影像。
4. 在 測試資料集詳細資料 的區段中，選擇 從電腦上傳影像。
5. 選擇 建立資料集。

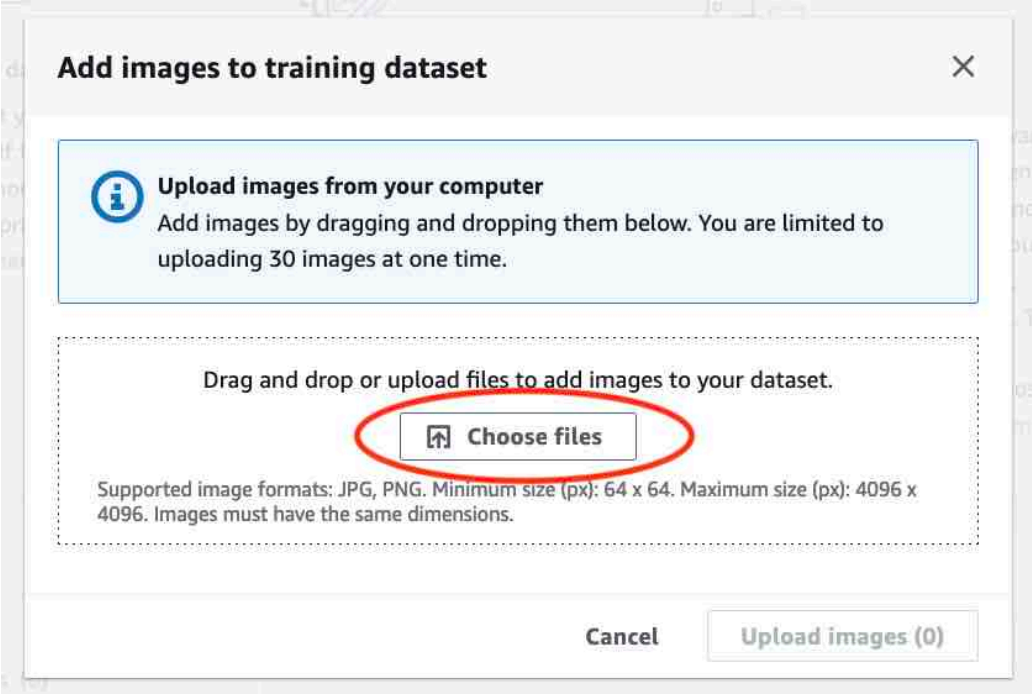


6. 隨即會顯示資料集頁面，其中包含對應資料集的培訓索引標籤和測試索引標籤。

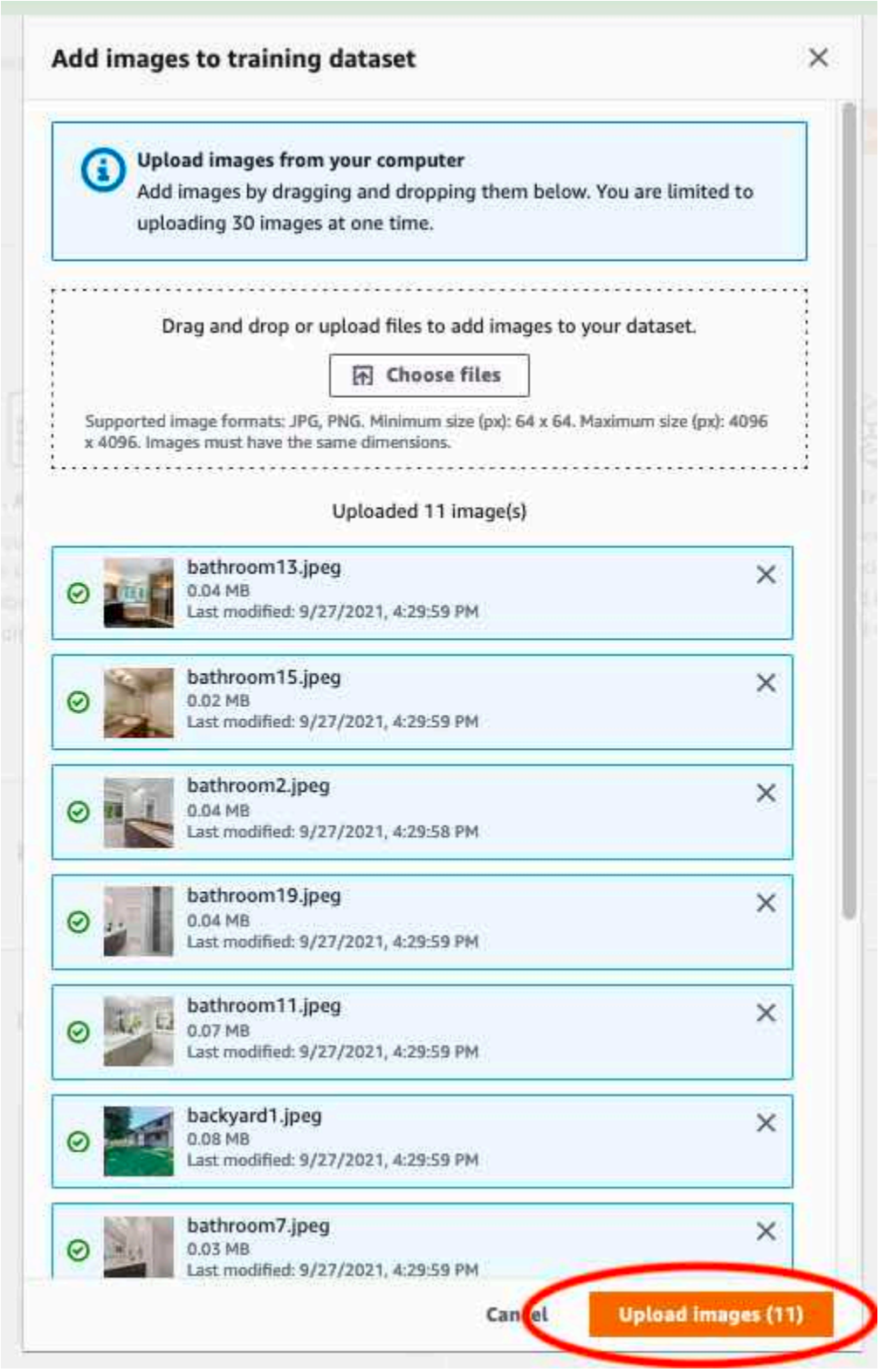
7. 在資料集頁面上，選擇 培訓 標籤。
8. 選擇 動作，然後選擇 新增影像至培訓資料集。



9. 在 新增影像至培訓資料集 的對話框中，選擇 選擇檔案。



10. 選擇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 上傳影像。Amazon Rekognition 自訂標籤可能需要幾秒鐘的時間才能將影像新增至資料集。



12. 如果您有更多影像要新增至培訓資料集中，請重複步驟 9-12。
13. 選擇測試標籤。

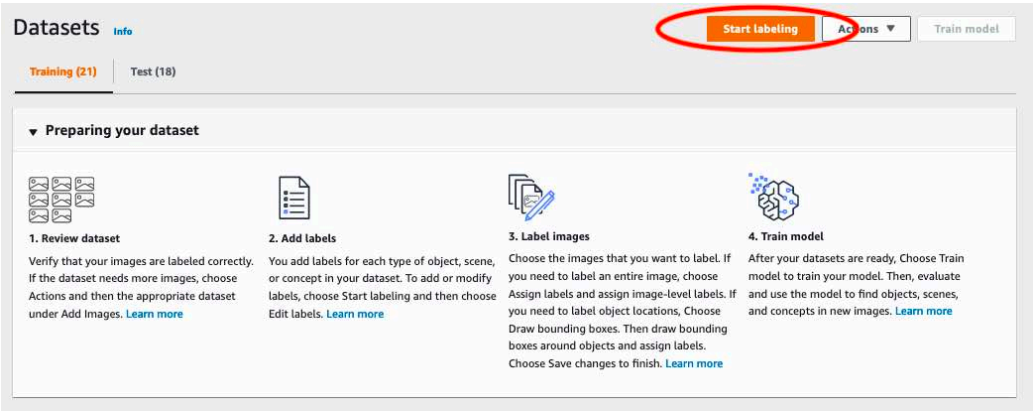
14. 重複步驟 8-12，將影像新增至測試資料集。在步驟 8 中，選擇 動作，然後選擇 新增影像至測試資料集。

步驟 5：新增標籤至專案

在此步驟中，您將為步驟 步驟 2：決定課程 中識別的每個類別新增一個標籤到專案中。

新增標籤 (主控台)

1. 在資料集庫頁面上，選擇 開始標籤，進入標籤模式。



2. 在資料集庫的 標籤 的區段中，選擇 編輯標籤 以開啟 管理標籤 的對話框。
3. 在編輯框中，輸入新標籤名稱。
4. 選擇 新增標籤。
5. 重複步驟 3 和 4，直至建立完所需的所有標籤為止。
6. 選擇 儲存 以儲存您新增的標籤。

步驟 6：為培訓和測試資料集指派影像層級標籤

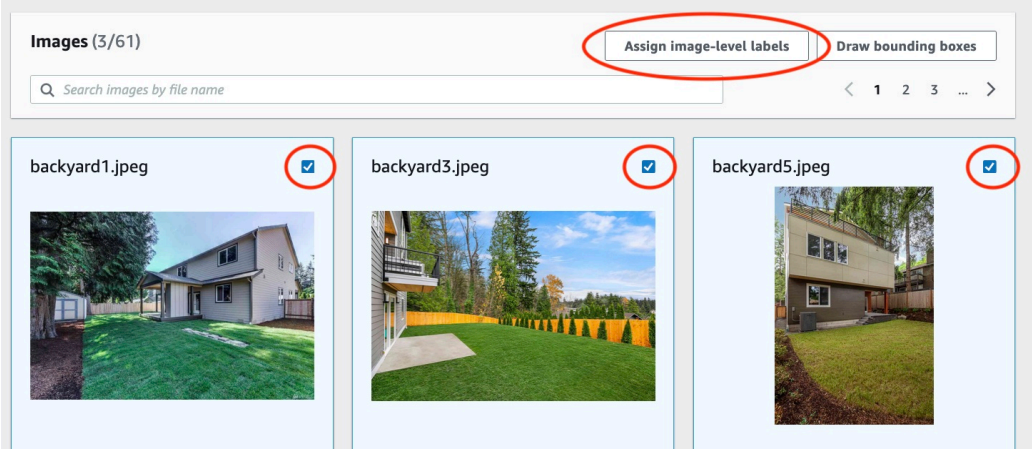
在此步驟中，您將為培訓和測試資料集中的每個影像指派單一影像層級。影像層級標籤是每個影像所代表的類別。

將影像層級標籤指派給影像 (主控台)

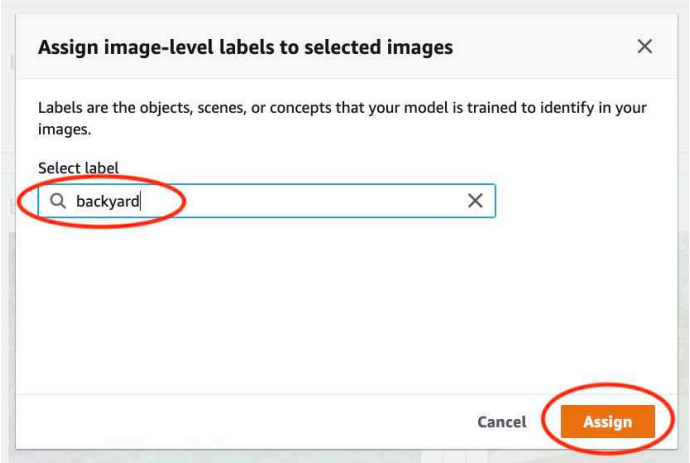
1. 在 資料集 的頁面上，選擇 培訓 標籤。
2. 選擇 開始標記 以進入標記模式。
3. 選擇要新增標籤的一張或多張影像。您一次只能選取單一頁面上的影像。若要在頁面上選取連續範圍的影像：

a. 選取第一個影像。
b. 按住 Shift 鍵。
c. 選取第二個影像。同時選取第一個影像和第二個影像之間的影像。
d. 放開 Shift 鍵。

4. 選擇 指派影像層級標籤。



5. 在 分配影像層級標籤至選取的影像 對話框中，選取要指派給一或多個影像的標籤。
6. 選擇 指派，為影像指派標籤。



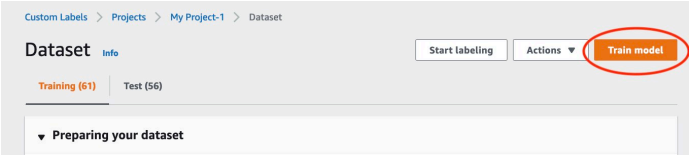
7. 重複標記，直至每個影像都用所需的標籤進行註釋。
8. 選擇測試標籤。
9. 重複步驟，將影像層級標籤指派給測試資料集影像。

步驟 7：培訓您的模型

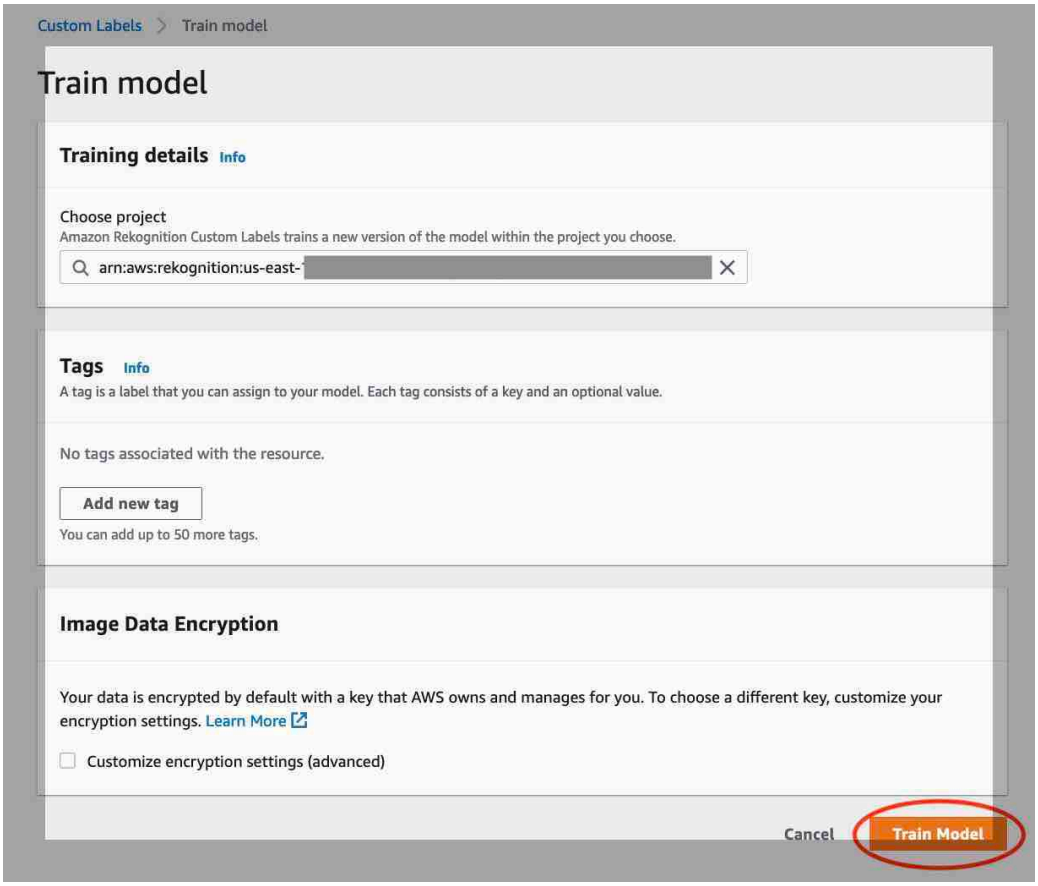
請使用以下步驟來訓練模型。如需詳細資訊，請參閱培訓 Amazon Rekognition 自訂標籤模型。

培訓您的模型（主控台）

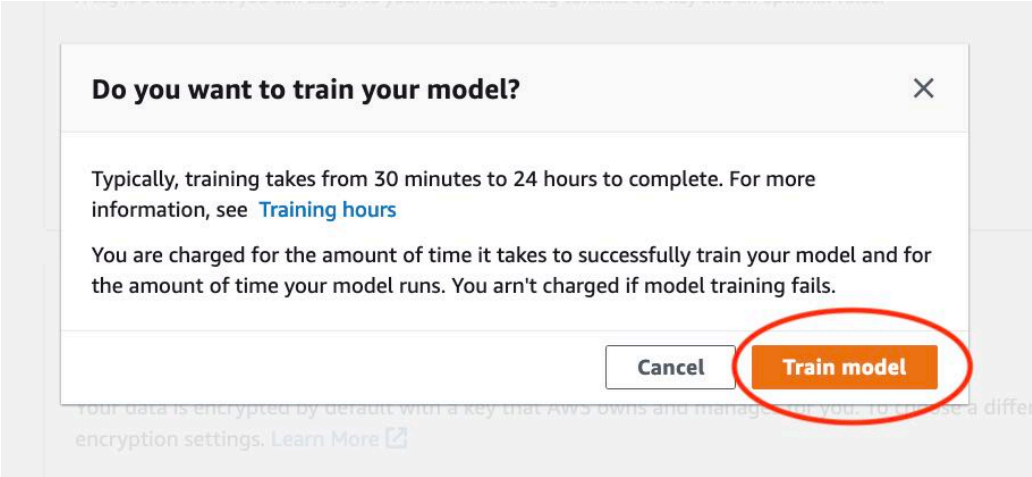
1. 在 資料集 頁面上，選擇 訓練模型。



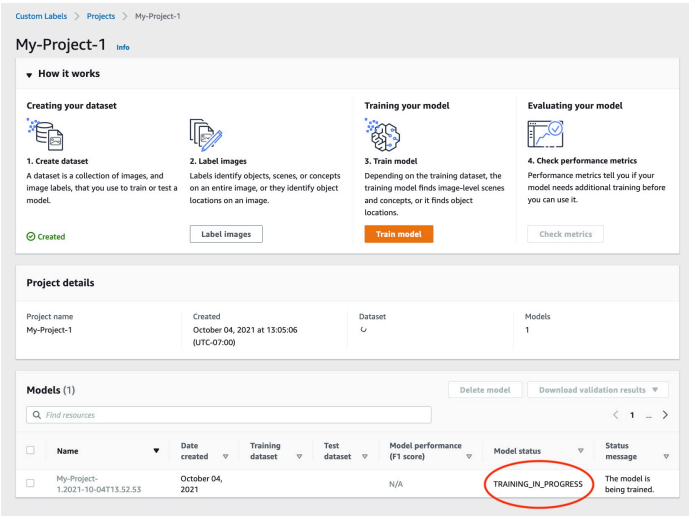
2. 在 培訓模型 的頁面上，選擇 培訓模型。專案的 Amazon Resource Name (ARN) 位於選擇專案的編輯框中。



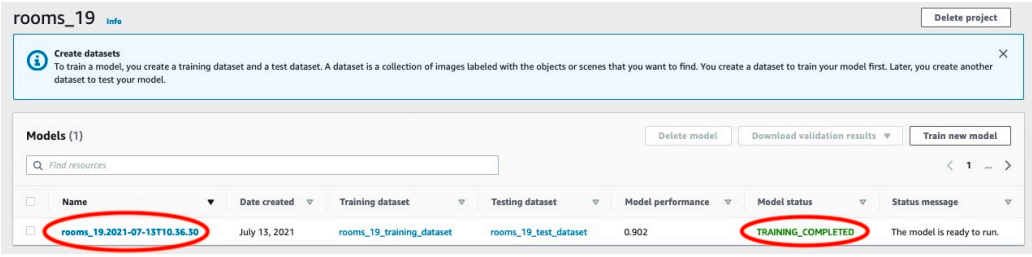
3. 在您是否要訓練模型？的對話框中，選擇訓練模型。



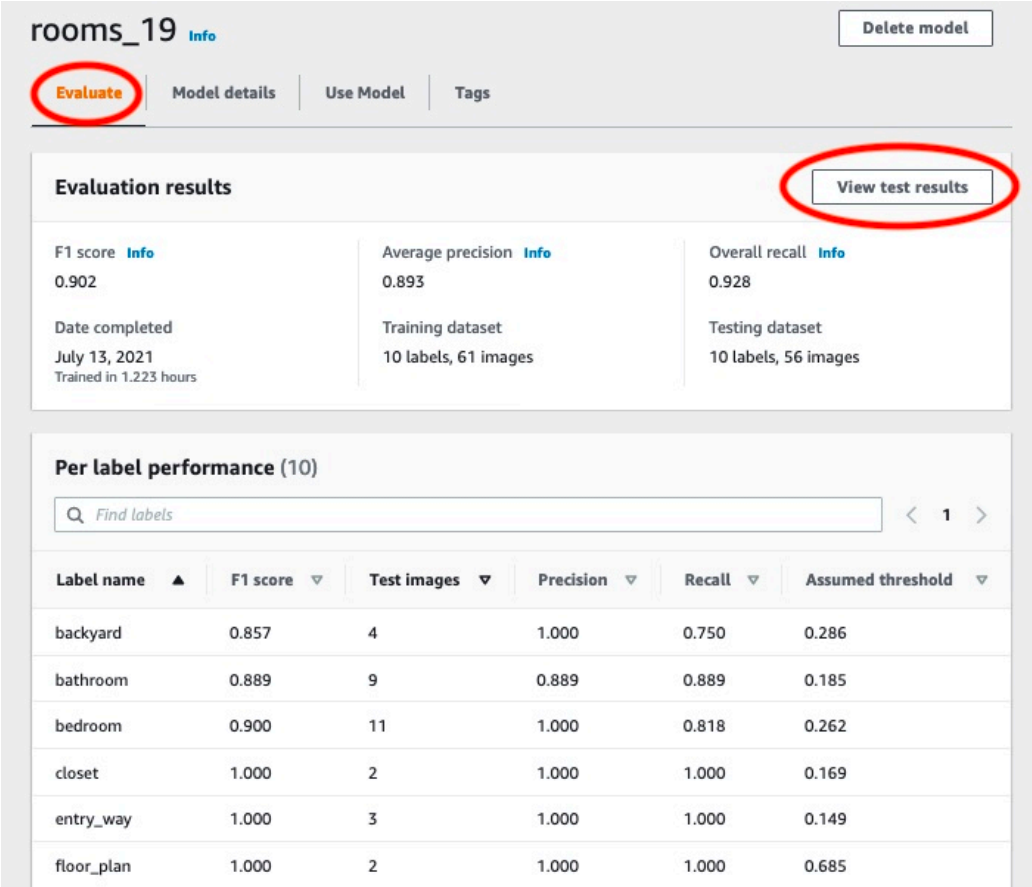
4. 在專案頁面的 模型 區段中，您可以看到培訓正在進行中。您可以透過查看模型版本的 Model Status 欄來檢查目前狀態。培訓模型需要一段時間才能完成。



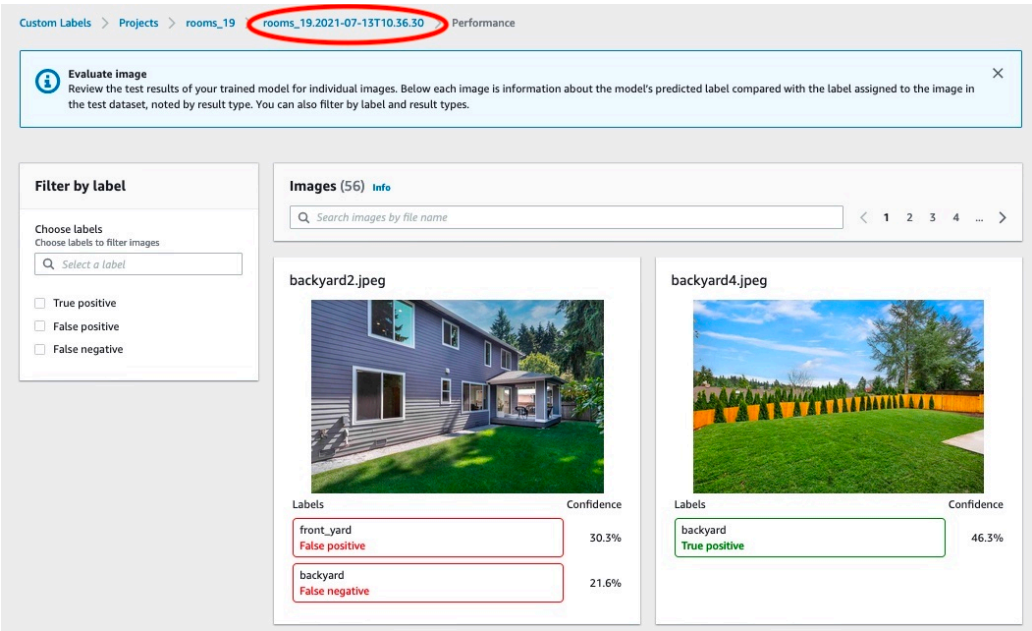
5. 訓練完成後，請選擇模型名稱。當模型狀態轉為訓練已完成時，表示訓練已經結束。



6. 選擇評估按鈕以查看評估結果。如需評估模型的資訊，請參閱 改善訓練過的 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，請參閱用於評估模型的指標。



8. 查看測試結果後，選擇模型名稱以返回模型頁面。



步驟 8：啟動模型

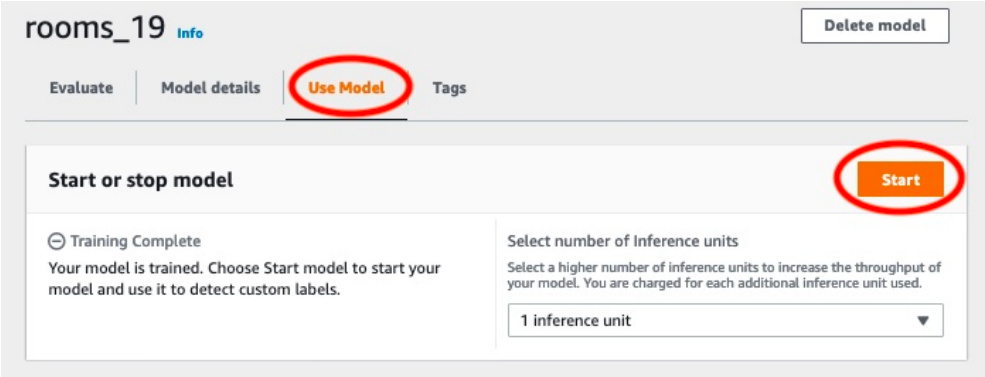
在此步驟中，您可以啟動模型。模型啟動後，您即可將其用於分析影像。

您需要根據模型執行時間付費。如果您不需要分析影像，請停止您的模型。您可以在稍後時間重新啟動您的模型。如需詳細資訊，請參閱執行培訓過的 Amazon Rekognition 自訂標籤模型。

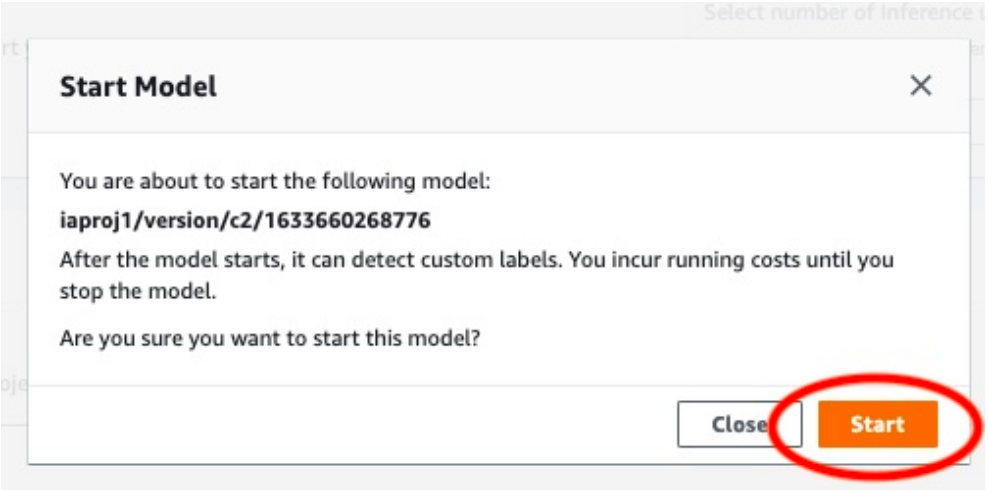
啟動模型

1. 選擇模型頁面上的使用模型索引標籤。
2. 在啟動或停止模型區段中，執行以下操作：

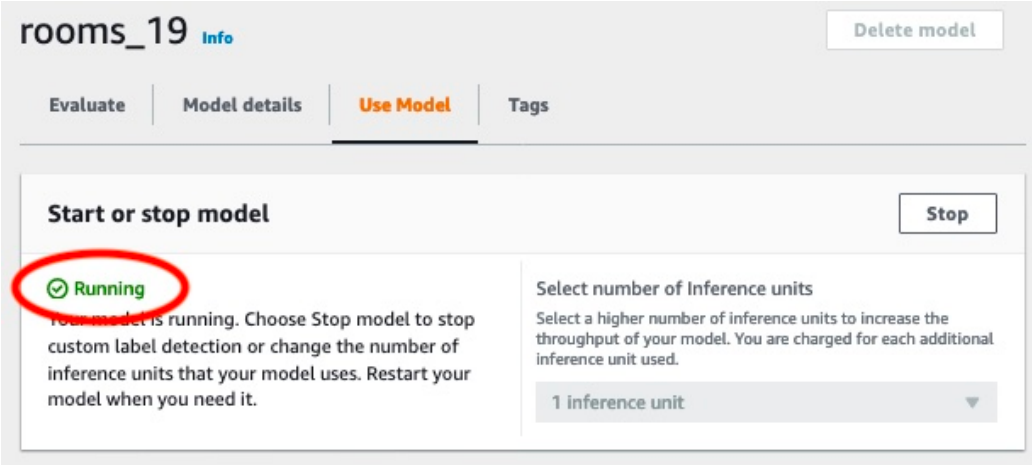
   a. 選擇 啟動。



   b. 在啟動模型的對話框中，選擇啟動。



3. 等待模型執行。當啟動或停止模型的區域中的狀態為執行中時，表示模型正在執行中。



步驟 9：使用模型分析影像

呼叫 DetectCustomLabels API 以分析影像。在此步驟中，您會使用 `detect-custom-labels` AWS Command Line Interface (AWS CLI) 命令來分析範例映像。您可以從 Amazon Rekognition 自訂標籤主控台取得 AWS CLI 命令。主控台會將 AWS CLI 命令設定為使用您的模型。您只需要提供儲存在 Amazon S3 儲存貯體中的影像。

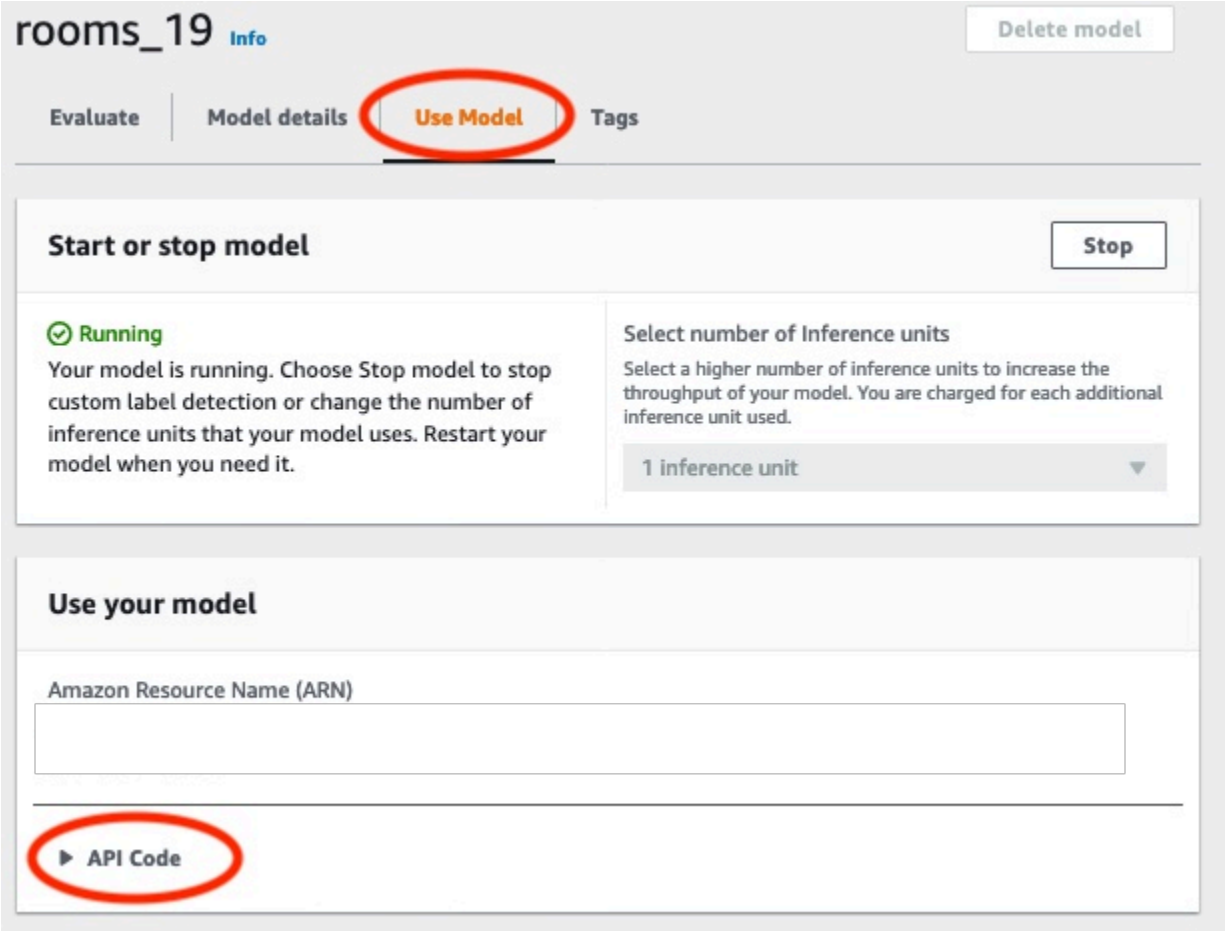
> **Note**
>
> 主控台還會提供 Python 範例程式碼。

來自 `detect-custom-labels` 的輸出包括在影像中找到的標籤清單、邊界框 (如果模型尋找物體位置)，以及模型對預測準確度的信賴度。

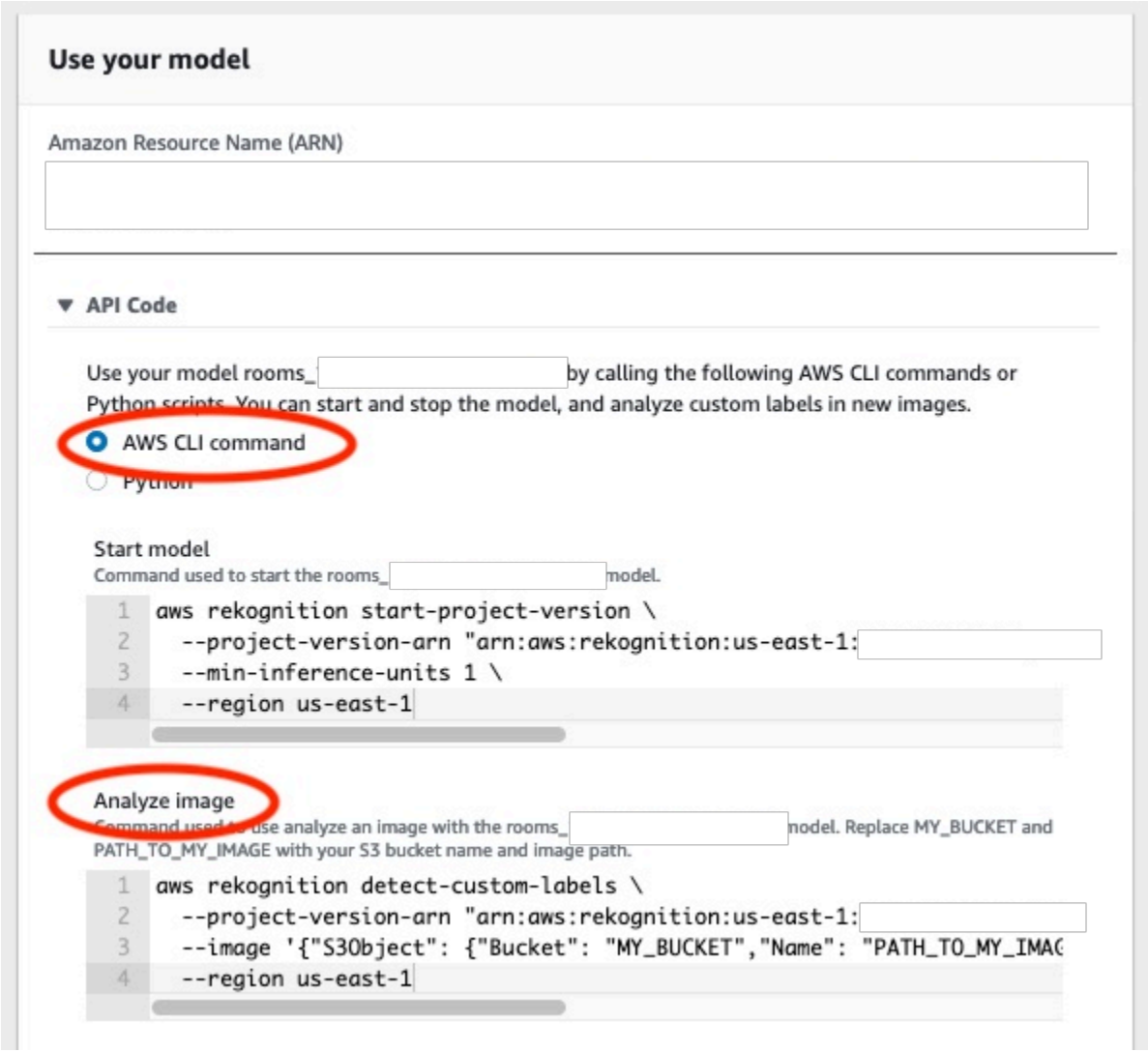
如需詳細資訊，請參閱使用經過培訓的模型分析圖像。

分析影像 (主控台)

1. 如果您尚未設定 ，請設定 AWS CLI。如需指示說明，請參閱 the section called “步驟 4：設定 AWS CLI 和 AWS SDKs”。
2. 選擇 使用模型 標籤，然後選擇 API 程式碼。



3. 選擇 AWS CLI 命令。
4. 在分析映像區段中，複製呼叫 的 AWS CLI 命令`detect-custom-labels`。



5. 將影像上傳至 Amazon S3 儲存貯體。如需指示說明，請參閱 Amazon 簡單儲存服務使用者指南中的 上傳物體至 Amazon S3。如果您正在使用 Rooms 專案中的影像，請使用您在 步驟 1：收集您的影像 中移至單獨資料夾的影像之一。

6. 在命令提示字元中，輸入您在上一個步驟中複製的 AWS CLI 命令。輸出應該如以下範例所示。

   `--project-version-arn` 的值應該是模型的 Amazon Resource Name (ARN)。`--region` 的值應該是您在其中建立模型的 AWS 區域。

   將 `MY_BUCKET` 和 `PATH_TO_MY_IMAGE` 變更為您在上一步驟中所使用的 Amazon S3 儲存貯體和影像。

   如果您是使用 自訂標籤存取 設定檔來取得憑證，請新增 `--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
```

該 AWS CLI 命令的 JSON 輸出看起來應如下列內容。Name 是模型找到的影像層級標籤的名稱。Confidence (0-100) 是模型對預測準確度的信賴度。

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

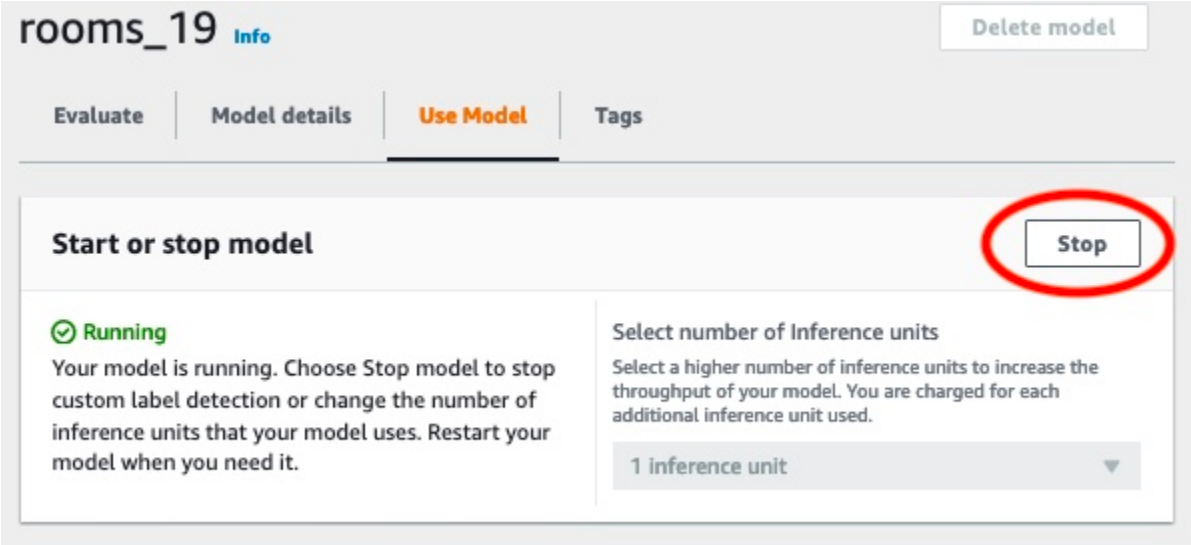
7. 繼續使用模型分析其他影像。如果不再使用，請停止模型。

步驟 10：停止模型

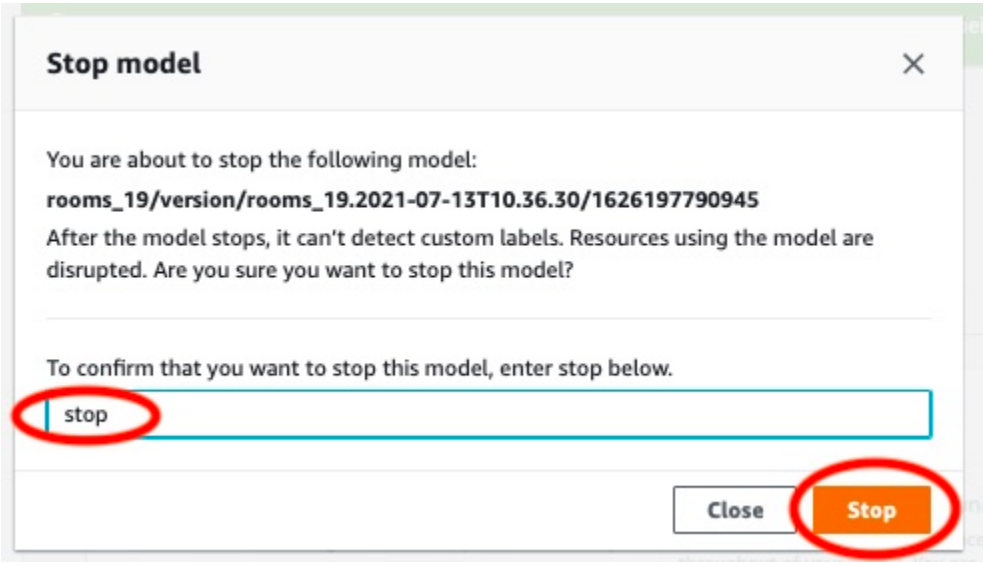
在此步驟中，您將停止執行模型。您需要根據模型執行時間付費。如果您已使用完成，請停止模型。

停止模型

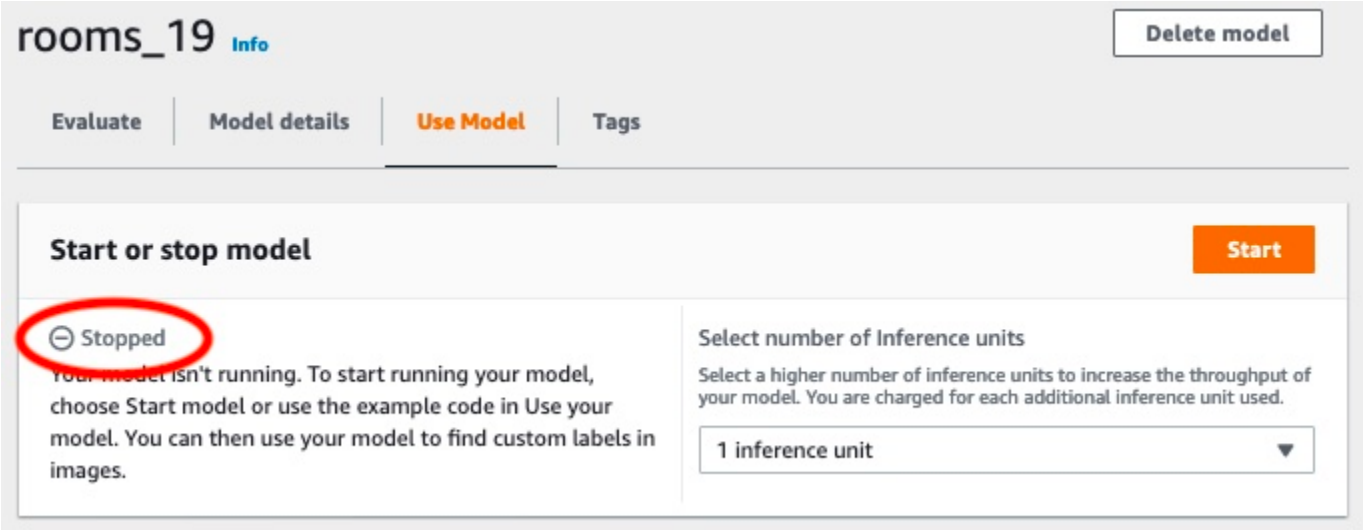
1. 在啟動或停止模型區段中，選擇停止。



2. 在停止模型對話框中，輸入 停止，以確認您要停止模型。



3. 選擇停止，以停止模型。當 啟動或停止模型 的區段中的狀態顯示為 已停止 時，模型即已停止。


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 自訂標籤模型

模型是由您訓練的軟體，用以尋找對您的業務具有獨特意義的概念、場景和物件。您可以使用 Amazon Rekognition 自訂標籤主控台或 AWS SDK 建立模型。在建立 Amazon Rekognition 自訂標籤模型前，我們建議您先閱讀 了解 Amazon Rekognition 自訂標籤。

本節提供有關使用主控台和 SDK 建立專案、為不同模型類型創建培訓和測試資料集以及培訓模型的資訊。後續章節將向您展示如何改進和使用您的模型。有關使用主控台建立和使用特定類型模型的教程，請參閱 分類映像。

主題

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

建立專案

專案用於管理模型版本、培訓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 建立專案。如需了解其他專案相關功能 (例如刪除專案)，請參閱 管理 Amazon Rekognition 自訂標籤專案。

您可以使用標籤來分類和管理 Amazon Rekognition 自訂標籤資源，包括您的專案。

CreateProject 操作可讓您在建立新專案時選擇性地指定標籤，提供標籤做為索引鍵/值對，供您用來分類和管理資源。

建立 Amazon Rekognition 自訂標籤專案 (主控台)

您可以使用 Amazon Rekognition 自訂標籤主控台建立專案。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 AWS 帳戶中建立 Amazon S3 儲存貯體 （主控台儲存貯體）。這個儲存貯體可用來存放您的專案。除非建立了主控台儲存貯體，否則您無法使用 Amazon Rekognition 自訂標籤主控台。

您可以使用 Amazon Rekognition 自訂標籤主控台建立專案。

建立專案 (主控台)

1. 登入 AWS 管理主控台 ，並在 https：//https://console.aws.amazon.com/rekognition/ 開啟 Amazon Rekognition 主控台。
2. 在左側視窗中，選擇 使用自訂標籤。畫面將會顯示 Amazon Rekognition 自訂標籤的登入頁面。
3. 在 Amazon Rekognition 自訂標籤登陸頁面中，選擇 開始使用。
4. 在左側視窗，選擇 專案。
5. 選擇 建立專案。
6. 在 專案名稱 中，輸入您的專案名稱。
7. 選擇 建立專案 以建立您的專案。
8. 請遵循 建立培訓和測試資料集 中的步驟，為您的專案建立培訓和測試資料集。

建立 Amazon Rekognition 自訂標籤專案 (SDK)

您可以透過呼叫 建立專案 來建立 Amazon Rekognition 自訂標籤專案。該回應是識別專案的 Amazon Resource Name (ARN)。建立專案後，您可以建立用於培訓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. 如果您尚未這麼做，請安裝並設定 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. 使用以下程式碼來建立專案。

   AWS CLI

   以下範例將建立一個專案並顯示其 ARN。

   將 `project-name` 的值變更為您要建立的專案的名稱。

   ```
   aws rekognition create-project --project-name my_project \
    --profile custom-labels-access --"CUSTOM_LABELS" --
   tags'{"key1":"value1","key2":"value2"}'
   ```

   Python

   以下範例將建立一個專案並顯示其 ARN。提供下列命令列參數：

- project_name – 您要建立的專案名稱。

```
# 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

以下範例將建立一個專案並顯示其 ARN。

提供以下命令列參數：

- project_name – 您要建立的專案名稱。

```
/*
   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.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. 請留意回應中顯示的專案 ARN 名稱。您將需要它來建立模型。
4. 請遵循 建立訓練和測試資料集 (SDK) 中的步驟，為您的專案建立培訓和測試資料集。

CreateProject 操作請求

以下是 CreateProject 操作請求的格式：

```
{
 "AutoUpdate": "string",
 "Feature": "string",
 "ProjectName": "string",
 "Tags": {
 "string": "string"
 }
}
```

建立培訓和測試資料集

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 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。如需更多詳細資訊，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。如果您使用的是 AWS 開發套件，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
- 準備影像
- 建立包含影像的訓練和測試資料集
- 標記檔案
- 偵錯資料集

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。

Image classification

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 Image classification。

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，請參閱多標籤影像分類。

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，請參閱從 Amazon S3 儲存貯體匯入映像。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

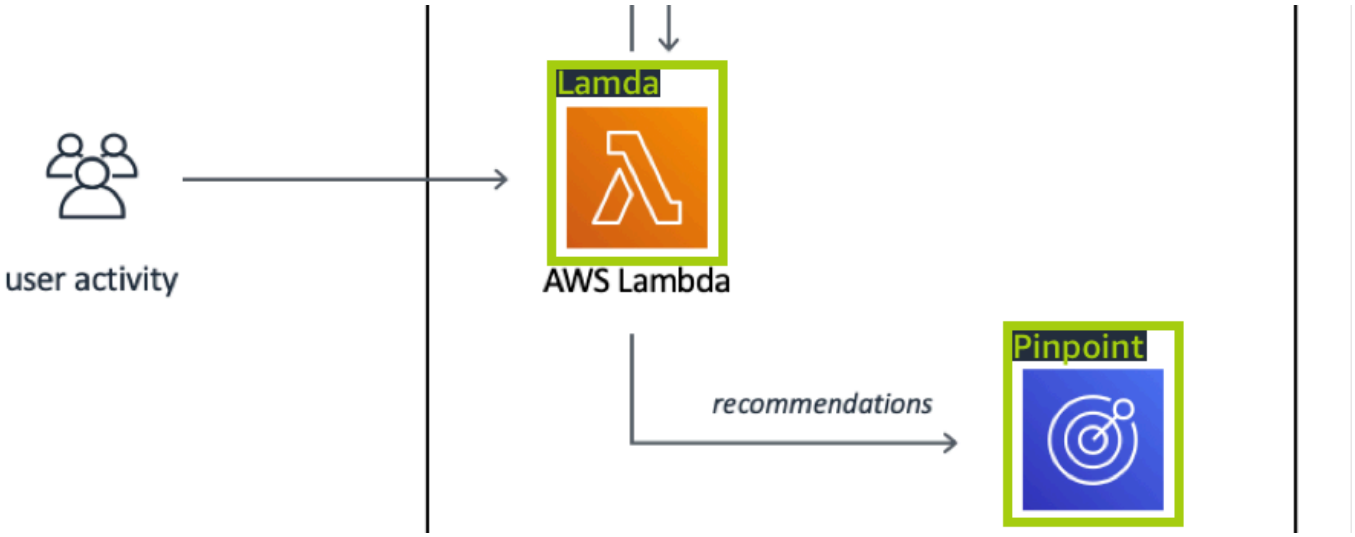
建立資料集時，您可以包含影像的週框方塊資訊。例如，您可以匯入包含週框方塊的 SageMaker AI Ground Truth 格式資訊清單檔案。您也可以在建立資料集之後新增週框方塊。如需詳細資訊，請參閱使用週框方塊標記物件。您可以根據需要新增標籤。如需詳細資訊，請參閱管理標籤。

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，下圖顯示結構圖，其中包含 AWS Lambda 和 Amazon Pinpoint 標誌周圍的標記週框方塊。



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。

品牌偵測 的範例專案展示了 Amazon Rekognition 自訂標籤如何使用標籤的邊界框來培訓尋找物體位置的模型。

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
Image classification	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 自訂標籤中的指南和配額。
- 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 自訂標籤會在培訓期間分割您的資料集，以便為您的專案建立培訓資料集 (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。為了完全控制培訓、測試和效能調整，我們建議您使用個別的培訓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：

- 從 Amazon S3 儲存貯體匯入映像
- 從本機電腦匯入映像
- 使用資訊清單檔案匯入映像
- 從現有資料集複製內容

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，從本機電腦匯入的影像即沒有標記。從 Amazon SageMaker AI 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 來建立訓練和測試資料集。

`CreateDataset` 操作可讓您在建立新資料集時選擇性地指定標籤，以便分類和管理 資源。

訓練資料集

您可以使用 AWS SDK 以下列方式建立訓練資料集。

- 將 CreateDataset 與您提供的 Amazon Sagemaker 格式清單檔案搭配使用。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 AI 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 使用現有的資料集建立資料集 (SDK)。
- 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 新增更多圖像 (SDK)。您需要先新增資料集項目，才能訓練模型。

測試資料集

您可以使用 AWS SDK 以下列方式建立測試資料集：

- 將 CreateDataset 與您提供的 Amazon Sagemaker 格式清單檔案搭配使用。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 AI 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 使用現有的資料集建立資料集 (SDK)。
- 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 新增更多圖像 (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，呼叫 DistributeDatasetEntries，將 20% 的訓練資料集項目移至測試資料集。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，請參閱分配培訓資料集 (SDK)。

從 Amazon S3 儲存貯體匯入映像

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 儲存貯體中的映像建立資料集。請改為建立參考影像來源位置的清單檔案。如需詳細資訊，請參閱使用資訊清單檔案匯入映像

在建立資料集期間，您可以選擇根據包含影像的資料夾名稱，為影像分配標籤名稱。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。同樣地，標籤 echo-dot 會被指派給資料夾 white-echo-dots 中的影像。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-bucket/alexa-devices/echo-dot，資料夾 white-echo-dot 中的影像會以 white-echo-dot 標記。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 AWS 區域中第一次開啟主控台時，使用 Amazon Rekognition 為您建立的 Amazon 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、分配標籤，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 管理主控台 ，並在 https：//https://console.aws.amazon.com/s3/ 開啟 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. 開啟 Amazon Rekognition 主控台：https://console.aws.amazon.com/rekognition/。
7. 選擇使用自訂標籤。
8. 選擇開始使用。
9. 在左側導覽視窗中，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 培訓模型（主控台） 中的步驟訓練模型。

從本機電腦匯入映像

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，請參閱從 Amazon S3 儲存貯體匯入映像。

> **Note**
>
> 您無法使用 AWS SDK 建立具有本機映像的資料集。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，請參閱使用資訊清單檔案匯入映像。

使用本機電腦 (主控台) 上的影像建立資料集

1. 開啟 Amazon Rekognition 主控台：https://console.aws.amazon.com/rekognition/。
2. 選擇使用自訂標籤。
3. 選擇開始使用。
4. 在左側導覽視窗中，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 培訓模型（主控台） 中的步驟訓練模型。

使用資訊清單檔案匯入映像

您可以使用 Amazon SageMaker AI Ground Truth 格式資訊清單檔案來建立資料集。您可以從 Amazon SageMaker AI Ground Truth 任務使用資訊清單檔案。如果您的影像和標籤不是 SageMaker AI Ground Truth 資訊清單檔案的格式，您可以建立 SageMaker AI 格式資訊清單檔案，並使用它來匯入已標記的影像。

`CreateDataset` 操作已更新，可讓您在建立新資料集時選擇性地指定標籤。標籤是可用於分類和管理 資源的鍵值組。

主題

- 使用 SageMaker AI Ground Truth 資訊清單檔案（主控台）建立資料集
- 使用 SageMaker AI Ground Truth 資訊清單檔案 (SDK) 建立資料集
- 建立資料集請求
- 使用 Amazon SageMaker AI Ground Truth 任務標記映像
- 建立清單檔案
- 在資訊清單檔案中匯入影像層級標籤
- 資訊清單檔案中的物件當地語系化
- 清單檔案的驗證規則
- 將其他資料集格式轉換為清單檔案

使用 SageMaker AI Ground Truth 資訊清單檔案（主控台）建立資料集

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

1. 執行下列其中一項操作，建立訓練資料集的清單檔案：

   - 遵循 中的指示，使用 SageMaker AI GroundTruth 任務建立資訊清單檔案使用 Amazon SageMaker AI Ground Truth 任務標記映像。
   - 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. 開啟 Amazon Rekognition 主控台：https://console.aws.amazon.com/rekognition/。
3. 選擇使用自訂標籤。
4. 選擇開始使用。
5. 在左側導覽視窗中，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訊區段中，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訊區段中，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訊區段中，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 培訓模型（主控台） 中的步驟訓練模型。

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

下列程序會說明如何使用 CreateDataset API 從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 AI Ground Truth 任務的輸出，或建立您自己的資訊清單檔案。

1. 如果您尚未這麼做，請安裝並設定 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. 執行下列其中一項操作，建立訓練資料集的清單檔案：
   - 遵循 中的指示，使用 SageMaker AI GroundTruth 任務建立資訊清單檔案使用 Amazon SageMaker AI Ground Truth 任務標記映像。
   - 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. 使用以下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
  --tags '{"key1": "value1", "key2": "value2"}'
```

Python

使用下列值建立資料集。請提供以下命令列參數：

- 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。請提供以下命令列參數：

- 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 培訓模型 (SDK) 中的步驟訓練模型。

建立資料集請求

以下是 CreateDataset 操作請求的格式：

```
{
"DatasetSource": {
"DatasetArn": "string",
"GroundTruthManifest": {
"S3Object": {
"Bucket": "string",
"Name": "string",
"Version": "string"
}
}
},
"DatasetType": "string",
"ProjectArn": "string",
"Tags": {
"string": "string"
}
```

```
}
```

使用 Amazon SageMaker AI Ground Truth 任務標記映像

透過 Amazon SageMaker AI Ground Truth，您可以使用來自您選擇的廠商公司 Amazon Mechanical Turk 的工作者，或是內部的私有人力資源，以及可讓您建立標籤組映像的機器學習。Amazon Rekognition 自訂標籤會從您指定的 Amazon S3 儲存貯體匯入 SageMaker AI Ground Truth 資訊清單檔案。

Amazon Rekognition 自訂標籤支援下列 SageMaker AI 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 AI Ground Truth 任務建立資訊清單檔案（主控台）

下列程序說明如何使用由 SageMaker AI Ground Truth 任務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。

使用 SageMaker AI Ground Truth 任務標記的影像建立資料集（主控台）

1. 登入 AWS 管理主控台，並在 https://console.aws.amazon.com/s3/：// 開啟 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**
   >
   > 主控台儲存貯體會在您第一次在 AWS 區域中開啟 Amazon Rekognition 自訂標籤主控台時建立。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. 開啟位在 https://console.aws.amazon.com/sagemaker/ 的 SageMaker AI 主控台。
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 進行資料標記。
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 AI Ground Truth 資訊清單檔案（主控台）建立資料集 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 AI 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 AI Ground Truth 任務。

建立清單檔案

您可以透過匯入 SageMaker AI Ground Truth 格式清單檔案來建立測試或訓練資料集。如果影像的標記格式不是 SageMaker AI Ground Truth 資訊清單檔案，請使用下列資訊來建立 SageMaker AI Ground Truth 格式資訊清單檔案。

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 自訂標籤支援具有 JSON 行的 SageMaker AI Ground Truth 資訊清單，格式如下：

- 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，請參閱在資訊清單檔案中匯入影像層級標籤。
- 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，請參閱資訊清單檔案中的物件當地語系化。

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

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。

使用 SageMaker AI 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 AI Ground Truth 格式清單檔案。如需詳細資訊，請參閱在資訊清單檔案中匯入影像層級標籤及資訊清單檔案中的物件當地語系化。

   > **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 AI Ground Truth 格式清單檔案。
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 AI Ground Truth 資訊清單檔案（主控台）建立資料集 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 AI Ground Truth 資訊清單檔案 (SDK) 建立資料集。

在資訊清單檔案中匯入影像層級標籤

若要匯入影像層級標籤（標記為不需要當地語系化資訊的場景、概念或物件的影像），請將 SageMaker AI Ground Truth 分類任務輸出格式 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，請參閱使用 SageMaker AI 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。

信賴度

(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
```

資訊清單檔案中的物件當地語系化

您可以將 SageMaker AI Ground Truth 邊界框任務輸出格式 JSON 行新增至資訊清單檔案，以匯入標示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，請參閱使用 SageMaker AI 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。

註釋

(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 AI Ground Truth 結構描述會強制執行語法驗證。如需更多詳細資訊，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

您可以使用以下資訊，從各種來源資料集格式建立 Amazon SageMaker AI 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，請參閱使用資訊清單檔案匯入映像。

主題

- 將 COCO 資料集轉換為資訊清單檔案格式
- 轉換多標籤 SageMaker AI Ground Truth 資訊清單檔案
- 從 CSV 檔案建立清單檔案。

將 COCO 資料集轉換為資訊清單檔案格式

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 資料集格式。

- `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釋清單 (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，請參閱資訊清單檔案中的物件當地語系化。

將 COCO 物件對應至自訂標籤 JSON Line

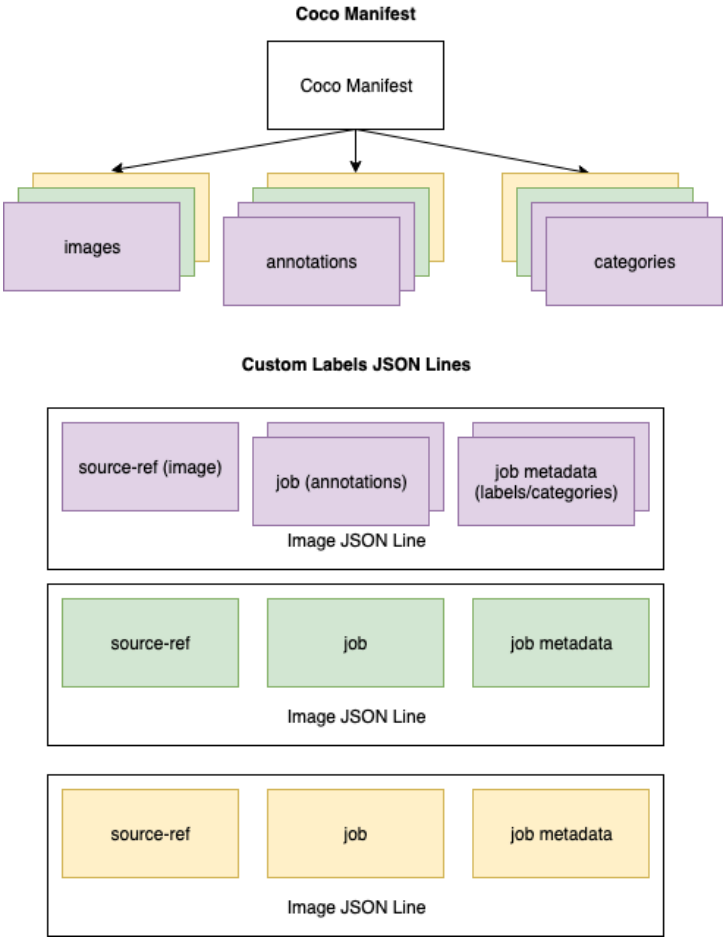
若要轉換 COCO 格式資料集，請將 COCO 資料集對應至 Amazon Rekognition 自訂標籤清單檔案，以進行物件本地化。如需詳細資訊，請參閱資訊清單檔案中的物件當地語系化。若要為每個影像建置 JSON Line，清單檔案需要對應 COCO 資料集 `image`、`annotation` 和 `category` 物件欄位 ID。

以下是 COCO 清單檔案範例。如需詳細資訊，請參閱COCO 資料集格式。

```
{
    "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。影像的每個 JSON 行都顯示來源參考、任務和任務中繼資料欄位。相符顏色表示單一影像的資訊。請注意，在資訊清單中，個別影像可能有多個註釋和中繼資料/類別。



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 資料集格式

COCO 資料集由五個區段的資訊組成，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。
- 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/
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"}
    ]
}
```

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/nnnnnnnnnnnnnnnnnnn.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. 安裝和設定 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 AI Ground Truth 資訊清單檔案（主控台）建立資料集 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 AI Ground Truth 資訊清單檔案 (SDK) 建立資料集。

轉換多標籤 SageMaker AI Ground Truth 資訊清單檔案

本主題說明如何將多標籤 Amazon SageMaker AI 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。

多標籤任務的 SageMaker AI 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 AI Ground Truth 任務的詳細資訊，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”。

取得 SageMaker AI Ground Truth 任務的資訊清單檔案

下列程序說明如何取得 Amazon SageMaker AI Ground Truth 任務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 AI Ground Truth 任務清單檔案

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 AI 資訊清單檔案

下列程序會從現有的多標籤格式 SageMaker AI GroundTruth 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。

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

轉換多標籤 SageMaker AI 資訊清單檔案

1. 使用以下 Python 程式碼。提供您在 取得 SageMaker AI Ground Truth 任務的資訊清單檔案 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 AI Ground Truth 資訊清單檔案（主控台）建立資料集 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 AI 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 多標籤影像分類。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 “在資訊清單檔案中匯入影像層級標籤”。

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 Image classification 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 多標籤影像分類 (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 AI Ground Truth 資訊清單檔案 （主控台） 建立資料集 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 AI Ground Truth 資訊清單檔案 (SDK) 建立資料集。

從現有資料集複製內容

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 使用現有的資料集建立資料集 (SDK)。

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

1. 開啟 Amazon Rekognition 主控台：https://console.aws.amazon.com/rekognition/。
2. 選擇使用自訂標籤。
3. 選擇開始使用。
4. 在左側導覽視窗中，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 培訓模型（主控台） 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)
- 變更和移除標籤 (主控台)

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. 開啟 Amazon Rekognition 主控台：https://console.aws.amazon.com/rekognition/。
2. 選擇使用自訂標籤。
3. 選擇開始使用。
4. 在左側導覽視窗中，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. 在編輯框中，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。

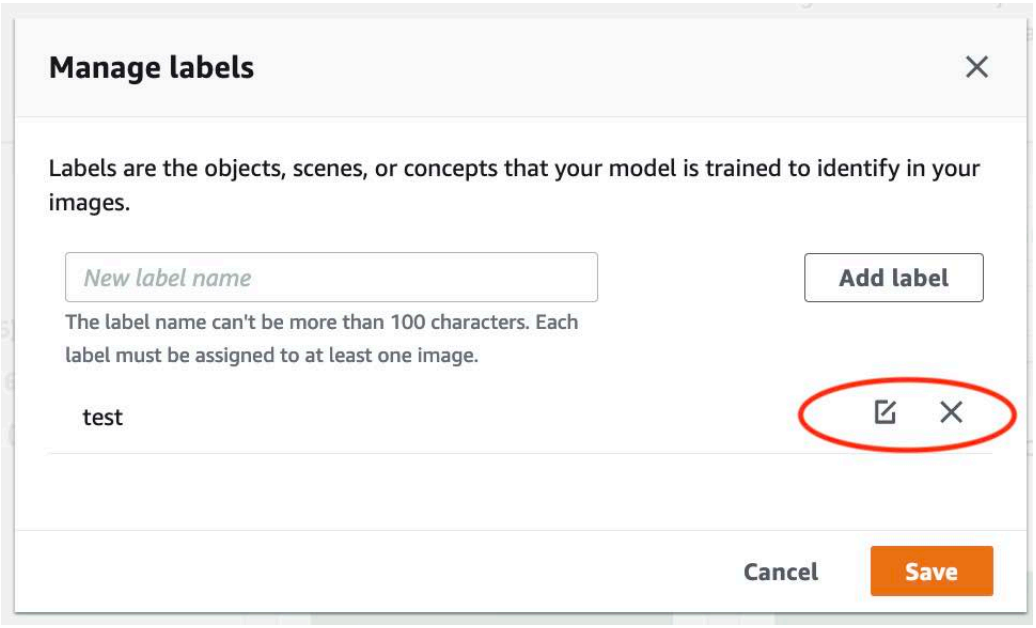
變更和移除標籤 (主控台)

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. 選擇開始使用。
4. 在左側導覽視窗中，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，請參閱新增更多圖像 (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 列出資料集條目 (SDK)。
2. 在檔案中，移除所有標籤註釋。如需詳細資訊，請參閱 在資訊清單檔案中匯入影像層級標籤 及 the section called “資訊清單檔案中的物件當地語系化”。
3. 使用檔案以透過 `UpdateDatasetEntries` API 更新資料集。如需詳細資訊，請參閱新增更多圖像 (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. 開啟 Amazon Rekognition 主控台：https://console.aws.amazon.com/rekognition/。
2. 選擇使用自訂標籤。
3. 選擇開始使用。
4. 在左側導覽視窗中，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 指派影像層級標籤。
11. 在將影像層級標籤指派給選取的影像對話方塊中，選取您要指派給影像或影像的標籤。
12. 選擇 指派，為影像指派標籤。
13. 重複標記，直至每個影像都用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 “在資訊清單檔案中匯入影像層級標籤”。

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，指定要為其分配標籤的影像的位置。如需詳細資訊，請參閱列出資料集條目 (SDK)。

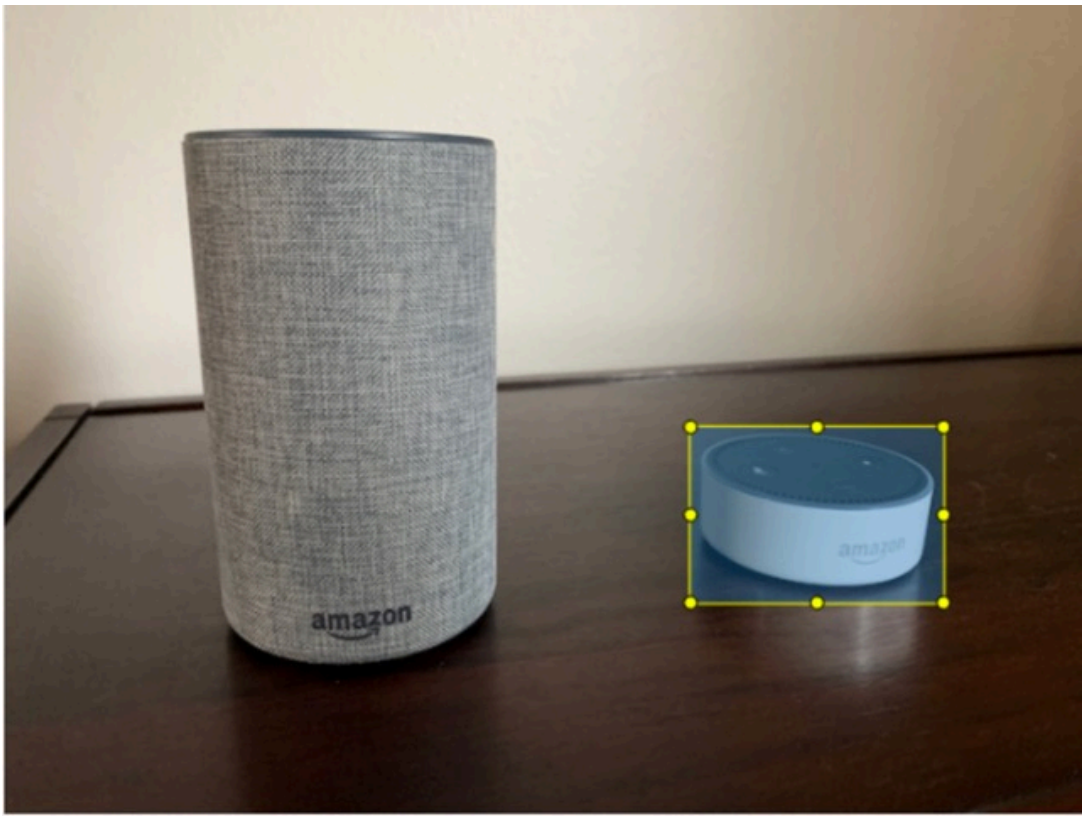
2. 使用 在資訊清單檔案中匯入影像層級標籤 中的資訊更新上一個步驟中傳回的 JSON Line。
3. 呼叫 `UpdateDatasetEntries` 以更新影像。如需詳細資訊，請參閱將更多圖像新增至資料集。

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。週框方塊是標示影像中物件的方塊。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。



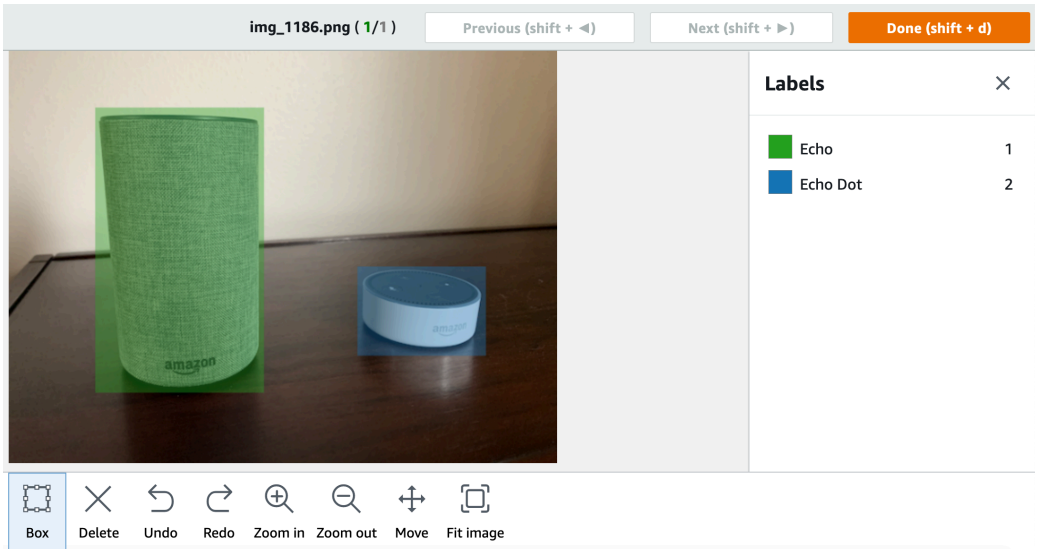
使用週框方塊尋找物件 (主控台)

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. 開啟 Amazon Rekognition 主控台： https://console.aws.amazon.com/rekognition/。
2. 選擇使用自訂標籤。
3. 選擇開始使用。
4. 在左側導覽視窗中，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 “資訊清單檔案中的物件當地語系化”。

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，請參閱列出資料集條目 (SDK)。
2. 使用 資訊清單檔案中的物件當地語系化 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，請參閱將更多圖像新增至資料集。

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

- 偵錯終端機資料集錯誤
- 偵錯非終端資料集錯誤

偵錯終端機資料集錯誤

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 自訂標籤中的指南和配額。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 自訂標籤中的指南和配額。

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，請參閱終端清單檔案內容錯誤。

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，則僅移除所需標籤的 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，請參閱資訊清單檔案中的物件當地語系化。

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。

偵錯非終端資料集錯誤

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，請參閱列出資料集條目 (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 Line 驗證錯誤。

- ERROR_NO_LABEL_ATTRIBUTES
- ERROR_INVALID_LABEL_ATTRIBUTE_FORMAT
- ERROR_INVALID_LABEL_ATTRIBUTE_METADATA_FORMAT
- ERROR_NO_VALID_LABEL_ATTRIBUTES
- ERROR_INVALID_BOUNDING_BOX
- ERROR_INVALID_IMAGE_DIMENSION
- ERROR_BOUNDING_BOX_TOO_SMALL

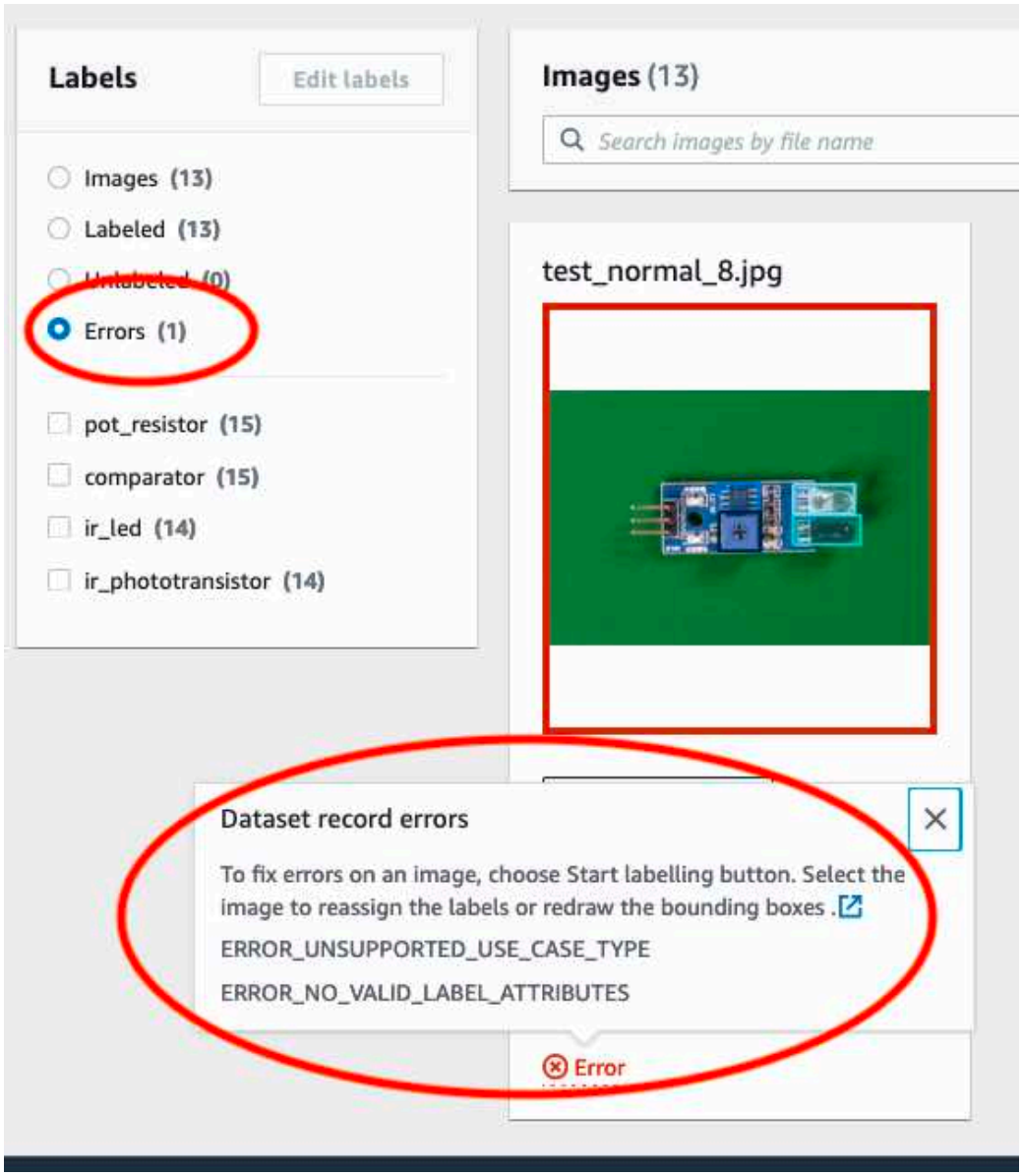
- ERROR_NO_VALID_ANNOTATIONS
- ERROR_MISSING_BOUNDING_BOX_CONFIDENCE
- ERROR_MISSING_CLASS_MAP_ID
- ERROR_TOO_MANY_BOUNDING_BOXES
- ERROR_UNSUPPORTED_USE_CASE_TYPE
- ERROR_INVALID_LABEL_NAME_LENGTH

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，請參閱列出資料集條目 (SDK)。

存取非終端錯誤 (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. 選擇開始使用。
4. 在左側導覽視窗中，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 Line 驗證錯誤 中的資訊來修正錯誤。



培訓 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 來培訓模型。如果模型培訓失敗，請使用中 偵錯失敗的模型訓練 的資訊來尋找失敗的原因。

> **Note**
>
> 您需根據成功培訓模型所需的時間付費。通常培訓需要 30 分鐘至 24 小時才能完成。如需更多詳細資訊，請參閱 培訓時間。

每次培訓模型時都會建立一個模型的新版本。Amazon Rekognition 自訂標籤會為模型建立一個名稱，該名稱是專案名稱和建立模型時的時間戳記的組合。

為了培訓您的模型，Amazon Rekognition 自訂標籤會製作來源培訓和測試圖像的副本。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，則需要擁有該 KMS 金鑰的以下權限。

- kms:創建權限
- kms:描述金鑰

如需更多詳細資訊，請參閱 AWS 金鑰管理服務概念。您的來源圖像不會受到影響。

您可以使用 KMS 伺服器端加密 (SSE-KMS) 來加密 Amazon S3 儲存貯體中的培訓和測試圖像，然後再由 Amazon Rekognition 自訂標籤進行複製。若要允許 Amazon Rekognition 自訂標籤存取您的映像， AWS 您的帳戶需要 KMS 金鑰的下列許可。

- kms:產生資料金鑰
- kms:解密

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

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

如需了解其他模型工作 (例如標籤模型)，請參閱 管理 Amazon Rekognition 自訂標籤模型。

主題

- 培訓模型（主控台）
- 培訓模型 (SDK)

培訓模型（主控台）

您可以使用 Amazon Rekognition 自訂標籤主控台來培訓模型。

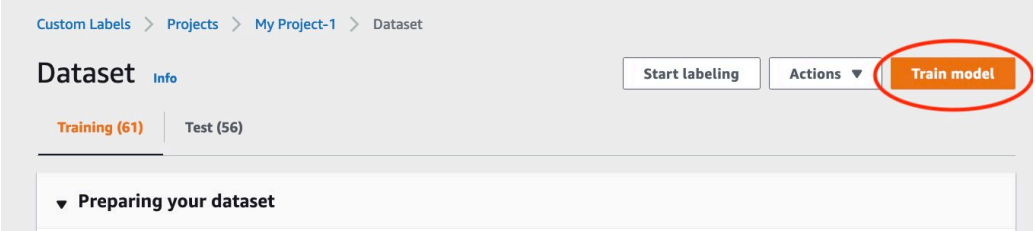
培訓需要具備培訓資料集和測試資料集的專案。如果您的專案沒有測試資料集，Amazon Rekognition 自訂標籤主控台會在培訓期間分割培訓資料集，以便為您的專案建立一個資料集。選擇的圖像是具代表

性的取樣，不會用於培訓資料集。我們建議您只有在沒有可使用的替代測試資料集時，才分割培訓資料集。分割培訓資料集會減少可用於培訓的圖像數量。

> **Note**
>
> 系統會根據培訓模型所需的時間來向您收取費用。如需更多詳細資訊，請參閱 培訓時間。

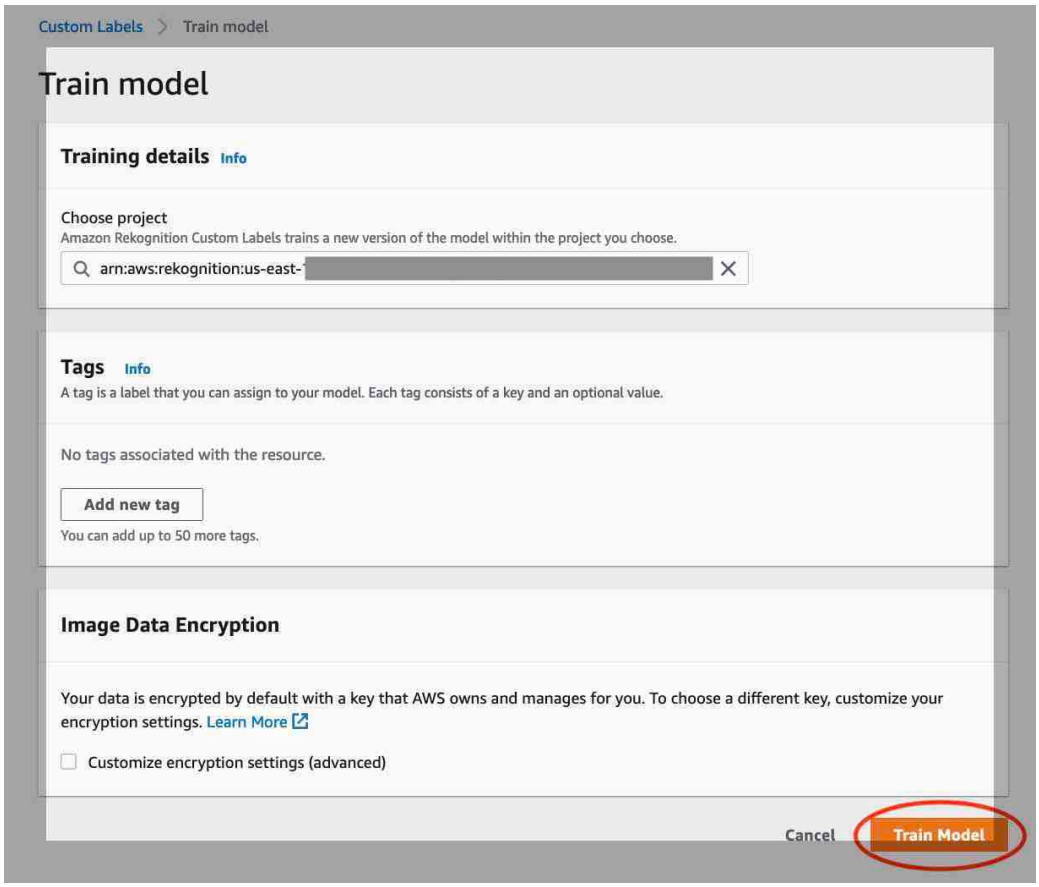
培訓您的模型（主控台）

1. 開啟 Amazon Rekognition 主控台：https://console.aws.amazon.com/rekognition/。
2. 選擇 使用自訂標籤。
3. 在左側導覽視窗中，選擇 專案。
4. 在 專案 頁面，選擇包含要培訓的模型的專案。
5. 在 專案 頁面上，選擇 培訓模型。

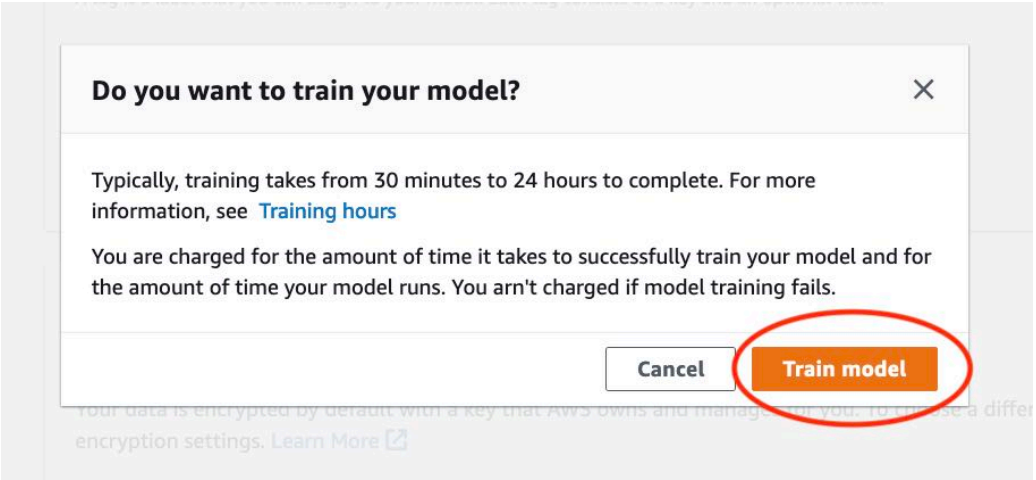


6. (可選) 如果您想要使用自己的 AWS KMS 加密金鑰，請執行以下操作：
   a. 在 圖像資料加密 中選擇 自訂加密配置 (進階)。
   b. 在 encryption.aws_kms_key，輸入您的金鑰的 Amazon Resource Name (ARN)，或選擇現有的 AWS KMS key。若要建立新的金鑰，請選擇 建立 AWS IMS 金鑰。
7. (可選) 如果您要新增標籤到模型，請執行以下操作：
   a. 在 標籤 區域，選擇 新增。
   b. 輸入下列資料：
      i. 金鑰 中的金鑰名稱。
      ii. 值 中的鍵/值。
   c. 若要新增更多標籤，請重複步驟 6a 和 6b。
   d. (選用) 如果您要移除標籤，請選擇要刪除的標籤旁邊的刪除。如果您要移除先前儲存的標籤，則會在您儲存變更時移除該標籤。

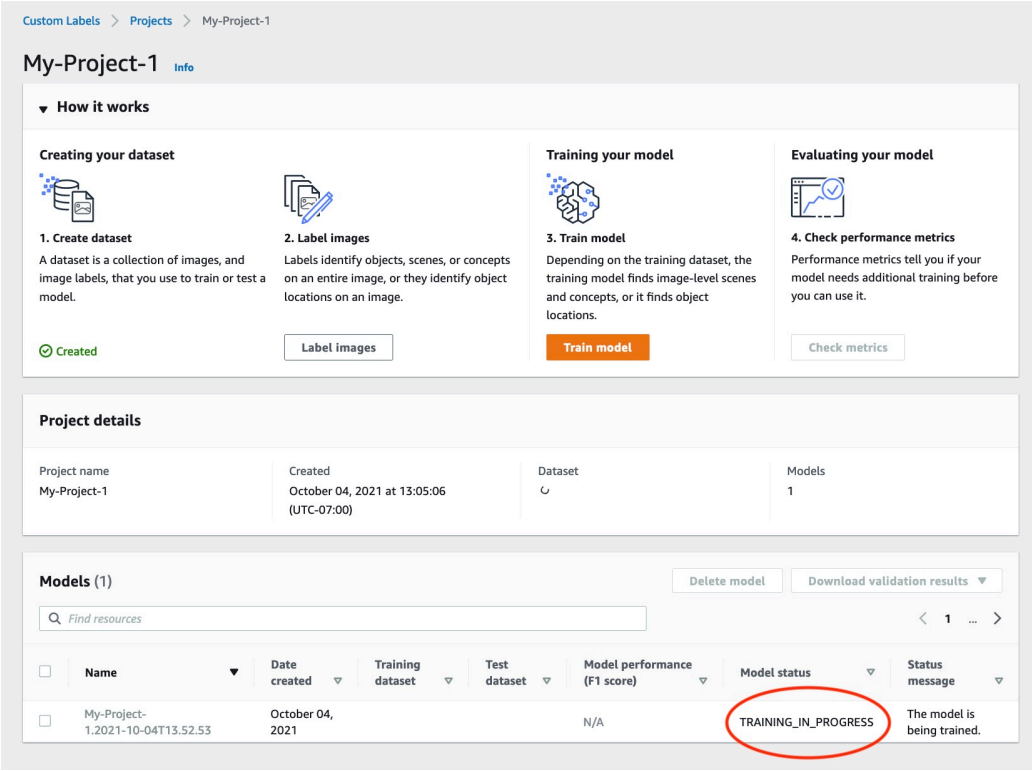
8. 在 培訓模型 的頁面上，選擇 培訓模型。專案的 Amazon Resource Name (ARN) 應該在 選擇專案 的編輯框中。如果沒有，請輸入專案的 ARN。



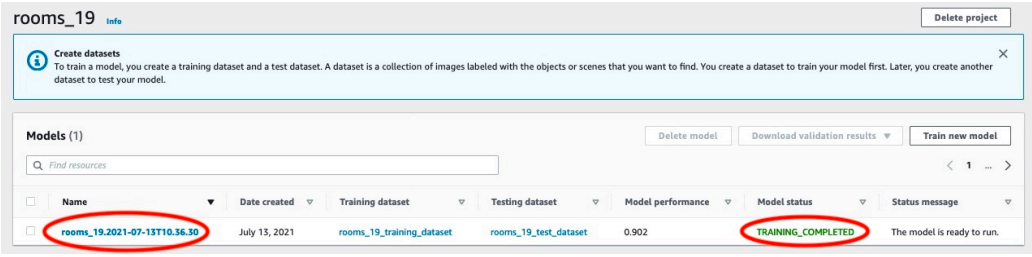
9. 在您是否要訓練模型？的對話框中，選擇訓練模型。



10. 在專案頁面的 模型 區域中，您可以在培訓正在進行的 `Model Status`欄位中檢查目前狀態。培訓模型需要一段時間才能完成。



11. 訓練完成後，請選擇模型名稱。當模型狀態轉為 培訓_完成 時，即培訓已經完成。如果培訓失敗，請參閱 偵錯失敗的模型訓練。



12. 下一步：評估您的模型。如需更多詳細資訊，請參閱 改善訓練過的 Amazon Rekognition 自訂標籤模型。

培訓模型 (SDK)

您可以呼叫 建立專案版本 來培訓模型。若要培訓模型，需要以下資料：

- 名稱 – 模型版本的唯一名稱。
- 專案 ARN — 管理模型的專案的 Amazon Resource Name (ARN)。
- 培訓結果位置 — 放置結果的 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，因為這樣可讓您設定權限，並避免與使用 Amazon Rekognition 自訂標籤主控台的培訓輸出發生潛在的命名衝突。

培訓使用與專案相關的培訓和測試資料集。如需詳細資訊，請參閱管理資料集。

> **Note**
>
> 或者，您可以指定項目外部的培訓和測試資料集清單檔案。如果您在使用外部清單檔案培訓模型後開啟主控台，Amazon Rekognition 自訂標籤將使用最後一組用於培訓的清單檔案為您建立資料集。您無法再透過指定外部資訊清單檔案來培訓專案的模型版本。如需更多詳細資訊，請參閱 建立專案版本。

來自 `CreateProjectVersion` 的回應是一個 ARN，您可以使用它來識別後續請求中的模型版本。您也可以使用 ARN 來保護模型版本。如需詳細資訊，請參閱保護 Amazon Rekognition 自訂標籤專案的安全。

培訓模型版本需要一段時間才能完成。本主題中的 Python 和 Java 範例使用等待程式以等待培訓完成。等待程式是一種實用的程式方法，用於輪詢特定狀態發生。或者，您可以通過呼叫 `DescribeProjectVersions` 來獲取培訓的當前狀態。當 `Status` 的欄位值轉為 `TRAINING_COMPLETED` 時，表示培訓完成。培訓完成後，您可以檢閱評估結果來評估模型的品質。

培訓模型 (SDK)

以下範例顯示如何使用與專案相關的培訓和測試資料集來培訓模型。

培訓模型 (SDK)

1. 如果您尚未這麼做，請安裝並設定 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. 使用下列範例程式碼來培訓專案。

   AWS CLI

   以下範例會建立一個模型。培訓資料集會被分割以建立測試資料集。取代以下項目：

   - `my_project_arn` 與專案的 Amazon Resource Name (ARN)。
   - `version_name` 取代為您選擇的唯一版本名稱。
   - `output_bucket` 取代為 Amazon Rekognition 自訂標籤儲存培訓結果的 Amazon S3 儲存貯體的名稱。
   - `output_folder` 取代為儲存培訓結果的資料夾名稱。
   - (可選參數) `--kms-key-id` 包含您的 AWS 金鑰管理服務客戶主金鑰的識別碼。

```
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

以下範例會建立一個模型。提供下列命令列參數：

- project_arn – 專案的 Amazon Resource Name (ARN)。
- version_name – 您所選模型的唯一版本名稱。
- output_bucket – Amazon Rekognition 自訂標籤儲存培訓結果的 Amazon S3 儲存貯體的名稱。
- output_folder – 儲存培訓結果的資料夾名稱。

您也可選擇性地提供以下命令列參數，以將標籤附加至您的模型：

- tag – 您選擇需要附加至模型的標籤名稱。
- 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

以下範例會培訓模型。提供下列命令列參數：

- project_arn – 專案的 Amazon Resource Name (ARN)。
- version_name – 您所選模型的唯一版本名稱。
- output_bucket – Amazon Rekognition 自訂標籤儲存培訓結果的 Amazon S3 儲存貯體的名稱。
- output_folder – 儲存培訓結果的資料夾名稱。

```
/*
   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. 如果培訓失敗，請參閱 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以及 DescribeProjectVersions 的回應中回報訓練錯誤。

錯誤可能是終端 (訓練無法繼續)，或者是非終端 (訓練可以繼續)。如需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 (清單檔案摘要以及訓練與測試驗證清單檔案)。使用驗證結果中的錯誤代碼來尋找本區段中的進一步資訊。本區段還會提供清單檔案錯誤 (在驗證清單檔案內容之前發生的終端錯誤) 的資訊。

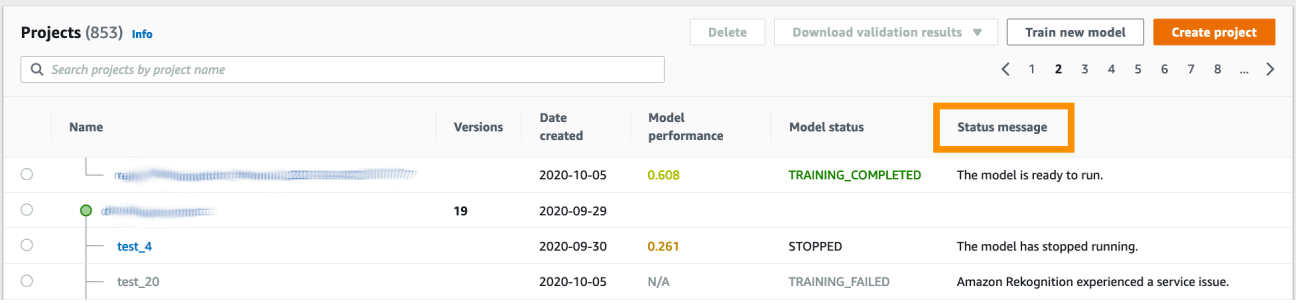
> **Note**
>
> 清單檔案是用於存放資料集內容的檔案。

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

終端錯誤

終端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、清單檔案錯誤和清單檔案內容錯誤。

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錯誤。專案管理儀表板會顯示專案清單，其中包含名稱、版本、建立日期、模型效能，以及指出模型狀態的狀態訊息，例如訓練已完成或失敗



如果您使用 AWS SDK，您可以檢查 DescribeProjectVersions 的回應，以查明是否已發生終端機資訊清單檔案錯誤或終端機資訊清單內容錯誤。在此情況下，`Status` 值為 `TRAINING_FAILED`，而 `StatusMessage` 欄位會包含錯誤。

服務錯誤

當 Amazon Rekognition 遇到服務問題且無法繼續訓練時，就會發生終端服務錯誤。例如，Amazon Rekognition 自訂標籤所依賴之另一項服務的失敗。Amazon Rekognition 遇到服務問題時，Amazon Rekognition 自訂標籤會在主控台中回報服務錯誤。如果您使用 AWS SDK，CreateProjectVersion 和 DescribeProjectVersions 會將訓練期間發生的服務錯誤引發為`InternalServerError`例外狀況。

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

終端機資訊清單檔案錯誤清單

清單檔案錯誤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錯誤。在驗證訓練和測試資料集的內容之前，即會偵測到清單檔案錯誤。清單檔案錯誤會防止回報非終端驗證錯誤。例如，空白的訓練清單檔案會產生清單檔案空白的錯誤。由於檔案空白，因此無法回報非終端 JSON Line 驗證錯誤。清單檔案摘要也不會建立。

您必須先修正清單檔案錯誤，才能訓練模型。

以下列出清單檔案錯誤。

- 清單檔案副檔名或內容無效。
- 清單檔案空白。
- 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 儲存貯體權限不正確。

終端機資訊清單內容錯誤清單

清單檔案內容錯誤指和清單檔案中的內容相關的終端錯誤。例如，如果您收到錯誤清單檔案中每個標籤所包含的標籤影像不足以執行自動分割，訓練即無法完成，因為訓練資料集中沒有足夠的已標記影像，因此無法建立測試資料集。

除了在主控台和 `DescribeProjectVersions` 的回應中回報之外，在清單檔案摘要中也能回報該錯誤以及任何其他終端清單檔案內容錯誤。如需詳細資訊，請參閱了解清單檔案摘要。

非終端 JSON Line 錯誤也會在單獨的訓練和測試驗證結果清單檔案中回報。Amazon Rekognition 自訂標籤所找到的非終端 JSON Line 錯誤不一定會和停止訓練的清單檔案內容錯誤有關。如需詳細資訊，請參閱了解培訓和測試驗證結果清單檔案。

您必須先修正清單檔案內容錯誤，才能訓練模型。

以下是清單檔案內容錯誤的錯誤訊息。

- 清單檔案包含太多無效資料列。
- 清單檔案包含來自多個 S3 儲存貯體的影像。
- 影像 S3 儲存貯體的擁有者 ID 無效。
- 清單檔案中每個標籤所包含的已標記影像不足以執行自動分割。
- 清單檔案的標籤太少。
- 清單檔案的標籤太多。
- 訓練和測試清單檔案之間的標籤重疊小於 {}%。
- 清單檔案可用標籤太少。
- 訓練和測試清單檔案之間的可用標籤重疊小於 {}%。
- 無法從 S3 儲存貯體複製影像。

非終端 JSON 行驗證錯誤的清單

JSON Line 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 Line 驗證錯誤不會在主控台中顯示。

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清單檔案中回報。Amazon Rekognition 自訂標籤會使用清單檔案中

的其他有效 JSON Lines 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檔案。如需驗證規則的資訊，請參閱 清單檔案的驗證規則。

> **Note**
>
> 如果 JSON Line 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，因為這些錯誤可能會造成未來錯誤或影響您的模型訓練。

Amazon Rekognition 自訂標籤可能產生下列非終端 JSON Line 驗證錯誤。

- 缺少 source-ref 金鑰。
- source-ref 值的格式無效。
- 找不到標籤屬性。
- 標籤屬性 {} 的格式無效。
- 標籤 attributemetadata 的格式無效。
- 找不到有效的標籤屬性。
- 一或多個週框方塊缺少可信度值。
- 類別對應中缺少一或多個類別 ID。
- JSON Line 的格式無效。
- 影像無效。檢查 S3 路徑和/或影像屬性。
- 週框方塊具有離框值。
- 週框方塊的高度和寬度太小。
- 週框方塊超過允許的最大值。
- 找不到有效的註釋。

了解清單檔案摘要

此清單檔案摘要包含以下資訊。

- 驗證期間遇到的 終端機資訊清單內容錯誤清單 錯誤資訊。
- 訓練和測試資料集中 非終端 JSON 行驗證錯誤的清單 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 Lines 總數。

如果沒有 終端機資訊清單檔案錯誤清單，即會在訓練期間建立清單檔案摘要。若要取得清單檔案摘要檔案 (manifest_summary.json) 的位置，請參閱 取得驗證結果。

> **Note**
>
> 清單檔案摘要中不會回報服務錯誤和清單檔案錯誤。如需詳細資訊，請參閱終端錯誤。

如需特定清單檔案內容錯誤的資訊，請參閱 終端清單檔案內容錯誤。

清單檔案摘要檔案格式

清單檔案有 2 個區段：`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` — 在測試資料集中找到的統計資料和錯誤。

`errors` 陣列中的物件包含清單檔案內容錯誤的錯誤代碼和訊息。

`error_line_indices` 陣列包含具有錯誤的訓練或測試清單檔案中每個 JSON Line 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，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 ERROR_INSUFFICIENT_USABLE_LABEL_OVERLAP。

```
{
    "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.
```

```
        }
    ]
}
```

清單檔案摘要範例

下列範例是部分清單檔案摘要，會顯示終端清單檔案內容錯誤 (ERROR_TOO_MANY_INVALID_ROWS_IN_MANIFEST)。error_json_line_indices 陣列包含對應的訓練或測試驗證清單檔案中非終端 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檔案

在訓練期間，Amazon Rekognition 自訂標籤會建立驗證結果清單檔案，以保留非終端 JSON Line 錯誤。驗證結果清單檔案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，您可以存取驗證清單檔案。如需詳細資訊，請參閱取得驗證結果。Amazon Rekognition 自訂標籤也會建立清單檔案摘要，其中會包含 JSON Line 錯誤的概觀資訊，例如錯誤位置和 JSON Line 錯誤計數。如需詳細資訊，請參閱了解清單檔案摘要。

> **Note**
>
> 只有在沒有 終端機資訊清單檔案錯誤清單 的情況下，才會建立驗證結果 (訓練與測試驗證結果清單檔案和清單檔案摘要)。

清單檔案會包含資料集中每個影像的 JSON Lines。在驗證結果清單檔案中，JSON Line 錯誤資訊會新增到發生錯誤的 JSON Lines。

JSON Line 錯誤指和單一影像相關的非終結錯誤。非終端驗證錯誤可能會使整個 JSON Line 或只有一部分失效。例如，如果 JSON Line 中參考的影像不是 PNG 或 JPG 格式，即會發生 ERROR_INVALID_IMAGE 錯誤，而且整個 JSON Line 會從訓練中排除。使用其他有效的 JSON Lines 繼續訓練。

在 JSON Line 中，錯誤可能表示 JSON Line 仍可以用於訓練。例如，如果與標籤相關聯的四個週框方塊之一的左側值為負值，則仍會使用其他有效的週框方塊來訓練模型。針對無效的週框方塊 (ERROR_INVALID_BOUNDING_BOX) 傳回 JSON Line 錯誤資訊。在此範例中，錯誤資訊會新增至發生錯誤的 `annotation` 物件。

警告錯誤 (例如 WARNING_NO_ANNOTATIONS) 不會用於訓練，並在清單檔案摘要中會計為忽略的 JSON Lines (ignored_json_lines)。如需詳細資訊，請參閱了解清單檔案摘要。此外，忽略的 JSON Lines 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 Line 驗證錯誤。

> **Note**
>
> 如果資料驗證錯誤太多，則會停止訓練，並在清單檔案摘要中回報 ERROR_TOO_MANY_INVALID_ROWS_IN_MANIFEST 終端錯誤。

如需更正 JSON Line 錯誤的資訊，請參閱 修正訓練錯誤。

JSON Line 錯誤格式

Amazon Rekognition 自訂標籤會將非終端驗證錯誤資訊新增至影像層級和物件本地化格式 JSON Lines。如需詳細資訊，請參閱the section called “建立清單檔案”。

影像層級錯誤

下面的範例會顯示影像層級 JSON Line 中的 Error 陣列。有兩組錯誤。與標籤屬性中繼資料 (在此範例中為 sport-metadata) 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)、錯誤訊息 (訊息)。如需詳細資訊，請參閱在資訊清單檔案中匯入影像層級標籤。

```
{
    "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

下面的範例會顯示物件本地化 JSON Line 中的錯誤陣列。JSON Line 會包含下列 JSON Line 區段中欄位的 `Errors` 陣列資訊。每個 `Error` 物件都包含錯誤代碼和錯誤訊息。

- 標籤屬性 — 標籤屬性欄位的錯誤。請參閱範例中的 `bounding-box`。
- 註釋 — 註釋錯誤 (週框方塊) 存放在標籤屬性內的 `annotations` 陣列中。
- 標籤屬性中繼資料 — 標籤屬性中繼資料的錯誤。請參閱範例中的 `bounding-box-metadata`。
- 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，請參閱資訊清單檔案中的物件當地語系化。

```
{
    "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 Line 錯誤的範例

下列物件本地化 JSON Line (格式化以供讀取) 顯示 ERROR_BOUNDING_BOX_TOO_SMALL 錯誤。在此範例中，週框方塊的尺寸 (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 終端機資訊清單內容錯誤清單 和 非終端 JSON 行驗證錯誤的清單 的錯誤資訊。有三個驗證結果檔案。

- training_manifest_with_validation.json — 新增了 JSON Line 錯誤資訊的訓練資料集清單檔案複本。
- training_manifest_with_validation.json — 新增了 JSON Line 錯誤資訊的測試資料集清單檔案複本。
- manifest_summary.json — 在訓練和測試資料集中找到的清單檔案內容錯誤和 JSON Line 錯誤的摘要。如需詳細資訊，請參閱了解清單檔案摘要。

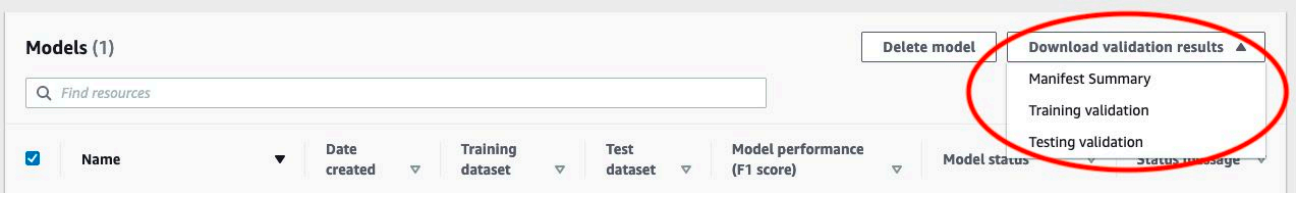
如需有關訓練和測試驗證清單檔案內容的資訊，請參閱 偵錯失敗的模型訓練。

> **Note**
>
> - 只有在訓練期間未產生 終端機資訊清單檔案錯誤清單 時，才會建立驗證結果。
> - 如果在驗證訓練和測試清單檔案之後發生服務錯誤，則會建立驗證結果，但 DescribeProjectVersions 的回應不包含驗證結果檔案位置。

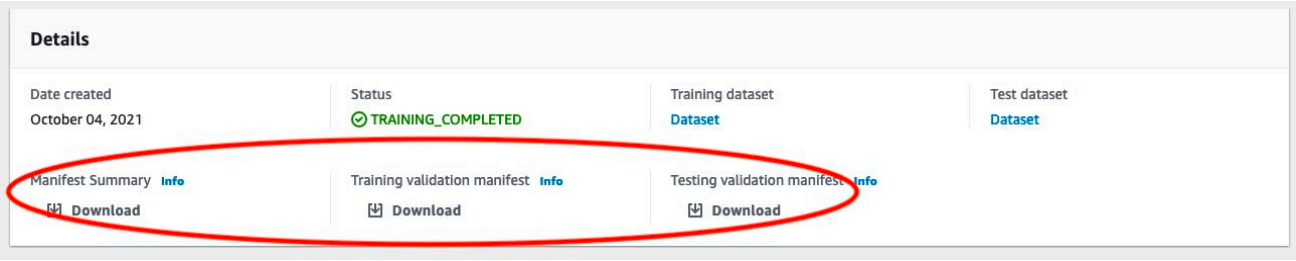
訓練完成或失敗後，您可以使用 Amazon Rekognition 自訂標籤主控台下載驗證結果，或呼叫 DescribeProjectVersions API 來取得 Amazon S3 儲存貯體位置。

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下載驗證結果，如下圖所示。模型面板顯示模型訓練和驗證結果，以及下載驗證結果的選項。



您也可以從模型的詳細資訊頁面存取下載驗證結果。詳細資訊頁面會顯示資料集詳細資訊，其中包含狀態、訓練和測試資料集，以及資訊清單摘要、訓練驗證資訊清單和測試驗證資訊清單的下載連結。



如需詳細資訊，請參閱培訓模型（主控台）。

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 DescribeProjectVersions API 來取得 S3 儲存貯體位置。若要訓練模型，請參閱 培訓模型 (SDK)。

針對訓練資料集 (TrainingDataResult) 和測試資料集 (TestingDataResult) 傳回 ValidationData 物件。清單檔案摘要會在 `ManifestSummary` 中傳回。

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 GetObject 操作。

取得驗證資料 (SDK)

1. 如果您尚未這麼做，請安裝並設定 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. 使用下列範例來取得驗證結果的位置。

Python

使用包含模型之專案的 Amazon Resource Name (ARN) 取代 `project_arn`。如需詳細資訊，請參閱管理 Amazon Rekognition 自訂標籤專案。使用模型版本的名稱取代 `version_name`。如需詳細資訊，請參閱培訓模型 (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。清單檔案摘要在 `ManifestSummary` 中。

修正訓練錯誤

您可以使用清單檔案摘要來識別訓練期間遇到的 終端機資訊清單內容錯誤清單 和 非終端 JSON 行驗證錯誤的清單。您必須修正清單檔案內容錯誤。建議您也修正非終端 JSON Line 錯誤。如需特定錯誤的資訊，請參閱 非終端 JSON Line 驗證錯誤 和 終端清單檔案內容錯誤。

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清單檔案中進行修正，並將其用於訓練模型。

完成修正後，您需要匯入更新的清單檔案並重新訓練模型。如需詳細資訊，請參閱建立清單檔案。

下列程序會說明如何使用清單檔案摘要修正終端清單檔案內容錯誤。此程序也會說明如何在訓練和測試驗證清單檔案中尋找及修正 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。檔案名稱為 training_manifest_with_validation.json、testing_manifest_with_validation.json 和 manifest_summary.json。如需詳細資訊，請參閱取得驗證結果。
2. 開啟清單檔案摘要檔案 (manifest_summary.json)。
3. 修正清單檔案摘要中的任何錯誤。如需詳細資訊，請參閱了解清單檔案摘要。
4. 在清單檔案摘要中，反覆執行 `training` 中的 `error_line_indices` 陣列並修正 `training_manifest_with_validation.json` 中在對應的 JSON Line 號碼的錯誤。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檔案”。
5. 反覆執行 `testing` 中的 `error_line_indices` 陣列並修正 `testing_manifest_with_validation.json` 中在對應的 JSON Line 號碼的錯誤。
6. 使用驗證清單檔案作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 “培訓模型”。

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 TrainingData 和 TestingData 輸入參數中驗證資料資訊清單檔案的位置至 CreateProjectVersion。如需詳細資訊，請參閱培訓模型 (SDK)。

JSON Line 錯誤優先順序

首先偵測到以下 JSON Line 錯誤。如果發生任何這些錯誤，JSON Line 錯誤的驗證即會停止。您必須先修正這些錯誤，才能修正任何其他 JSON Line 錯誤

- MISSING_SOURCE_REF
- ERROR_INVALID_SOURCE_REF_FORMAT
- ERROR_NO_LABEL_ATTRIBUTES
- ERROR_INVALID_LABEL_ATTRIBUTE_FORMAT
- ERROR_INVALID_LABEL_ATTRIBUTE_METADATA_FORMAT

- ERROR_MISSING_BOUNDING_BOX_CONFIDENCE
- ERROR_MISSING_CLASS_MAP_ID
- ERROR_INVALID_JSON_LINE

終端清單檔案錯誤

本主題會描述 終端機資訊清單檔案錯誤清單。清單檔案錯誤沒有關聯的錯誤代碼。當發生終端清單檔案錯誤時，不會建立驗證結果清單檔案。如需詳細資訊，請參閱了解清單檔案摘要。終端清單檔案錯誤會防止回報 非終端 JSON Line 驗證錯誤。

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。

修正錯誤清單檔案副檔名或內容無效。

- 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。

清單檔案空白。

訓練或用於訓練的測試清單檔案存在，但是空白的。對於您用於訓練和測試的每個影像，清單檔案都需要一個 JSON Line。

修正錯誤清單檔案空白。

1. 檢查哪些訓練或測試清單檔案空白。
2. 將 JSON Lines 新增到空白清單檔案。如需詳細資訊，請參閱建立清單檔案。或者，請使用主控台建立新資料集。如需詳細資訊，請參閱the section called “建立包含影像的資料集”。

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 自訂標籤中的指南和配額。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。

修正錯誤清單檔案大小超過支援的大小上限。

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。

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 S3 儲存貯體權限不正確。

- 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。

無法寫入至輸出 S3 儲存貯體。

服務無法產生訓練輸出檔案。

修正錯誤無法寫入至輸出 S3 儲存貯體。

- 檢查在 CreateProjectVersion 的 OutputConfig 輸入參數中的 Amazon S3 儲存貯體資訊是否正確。

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

終端清單檔案內容錯誤

本主題會描述清單檔案摘要中回報的 終端機資訊清單內容錯誤清單。清單檔案摘要包含每個偵測到之錯誤的錯誤代碼和訊息。如需詳細資訊，請參閱了解清單檔案摘要。終端清單檔案內容錯誤不會停止回報 非終端 JSON 行驗證錯誤的清單。

ERROR_TOO_MANY_INVALID_ROWS_IN_MANIFEST

錯誤訊息

清單檔案包含太多無效資料列。

其他資訊

如果有太多 JSON Lines 包含無效內容，即會發生 ERROR_TOO_MANY_INVALID_ROWS_IN_MANIFEST 錯誤。

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

修正 ERROR_TOO_MANY_INVALID_ROWS_IN_MANIFEST

1. 檢查 JSON Line 錯誤的清單檔案。如需詳細資訊，請參閱了解培訓和測試驗證結果清單檔案。
2. 修正發生錯誤的 JSON Lines 如需詳細資訊，請參閱 非終端 JSON Line 驗證錯誤。

ERROR_IMAGES_IN_MULTIPLE_S3_BUCKETS

錯誤訊息

清單檔案包含來自多個 S3 儲存貯體的影像。

其他資訊

清單檔案只能參考存放在單一儲存貯體中的影像。每個 JSON Line 都會以 source-ref 的值存放影像位置的 Amazon S3 位置。在下列範例中，儲存貯體名稱為 my-bucket。

```
"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。

修正 **ERROR_IMAGES_IN_MULTIPLE_S3_BUCKETS**

- 請確保所有影像都位於同一個 Amazon S3 儲存貯體中，且每個 JSON Line 中 source-ref 的值都會參考存放影像的儲存貯體。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 Lines。

ERROR_INVALID_PERMISSIONS_IMAGES_S3_BUCKET

錯誤訊息

影像 S3 儲存貯體的權限無效。

其他資訊

包含影像的 Amazon S3 儲存貯體的權限不正確。

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

修正 **ERROR_INVALID_PERMISSIONS_IMAGES_S3_BUCKET**

- 檢查包含影像的儲存貯體的權限。影像的 `source-ref` 值包含儲存貯體位置。

ERROR_INVALID_IMAGES_S3_BUCKET_OWNER

錯誤訊息

影像 S3 儲存貯體的擁有者 ID 無效。

其他資訊

包含訓練或測試影像之儲存貯體的擁有者與包含訓練或測試清單檔案之儲存貯體的擁有者不同。您可以使用下列命令來尋找儲存貯體的擁有者。

```
aws s3api get-bucket-acl --bucket amzn-s3-demo-bucket
```

`OWNER ID` 必須和存放影像和清單檔案的儲存貯體相符。

修正 ERROR_INVALID_IMAGES_S3_BUCKET_OWNER

1. 選擇所需的訓練、測試、輸出和影像儲存貯體的擁有者。擁有者必須擁有使用 Amazon Rekognition 自訂標籤的權限。
2. 針對目前並非由所需擁有者擁有的每個儲存貯體，建立由偏好的擁有者擁有的新 Amazon S3 儲存貯體。
3. 將舊儲存貯體內容複製到新儲存貯體。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。

ERROR_INSUFFICIENT_IMAGES_PER_LABEL_FOR_AUTOSPLIT

錯誤訊息

清單檔案中每個標籤所包含的已標記影像不足以執行自動分割。

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。當沒有足夠的影像來建立可接受的測試資料集時，就會發生 ERROR_INSUFFICIENT_IMAGES_PER_LABEL_FOR_AUTOSPLIT。

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

修正 ERROR_INSUFFICIENT_IMAGES_PER_LABEL_FOR_AUTOSPLIT

- 將更多已標記影像新增至訓練資料集。將影像新增至訓練資料集，或將 JSON Lines 新增至訓練清單檔案，即可在 Amazon Rekognition 自訂標籤主控台中新增影像。如需詳細資訊，請參閱管理資料集。

ERROR_MANIFEST_TOO_FEW_LABELS

錯誤訊息

清單檔案的標籤太少。

其他資訊

訓練和測試資料集具有所需的最小數量標籤。最小值會依據資料集是否訓練/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而定。如果分割訓練資料集以建立測試資料集，則資料集中的標籤數量會在訓練資料集分割後確定。如需詳細資訊，請參閱Amazon Rekognition 自訂標籤中的指南和配額。

修正 ERROR_MANIFEST_TOO_FEW_LABELS (主控台)

1. 新增更多新標籤至資料集。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。如果您的模型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”。

修正 ERROR_MANIFEST_TOO_FEW_LABELS (JSON Line)

- 為具有新標籤的新影像新增 JSON Lines。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，您可以新增新的標籤名稱至 `class-name` 欄位。例如，下列影像的標籤為 Sunrise。

```
{
    "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"
    }
}
```

如果模型偵測到物件位置，請新增新標籤至 `class-map`，如下列範例所示。

```
{
 "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"
 }
}
```

您需要將類別對應表對應至週框方塊註釋。如需詳細資訊，請參閱資訊清單檔案中的物件當地語系化。

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，則僅移除所需標籤的 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，請參閱資訊清單檔案中的物件當地語系化。

ERROR_INSUFFICIENT_LABEL_OVERLAP

錯誤訊息

訓練和測試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%。

修正 ERROR_INSUFFICIENT_LABEL_OVERLAP (主控台)

- 從訓練資料集中移除標籤。或者，新增更多常用標籤至測試資料集。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。

從訓練資料集 (JSON Line) 移除標籤，以修正 ERROR_INSUFFICIENT_LABEL_OVERLAP

- 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 Line。如果 JSON Line 包含多個標籤，則僅移除所需標籤的 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清單檔案中的每個 JSON Line 執行此動作。

  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，請參閱資訊清單檔案中的物件當地語系化。

新增通用標籤至測試資料集 (JSON Line) 以修正錯誤 ERROR_INSUFFICIENT_LABEL_OVERLAP

- 新增 JSON Lines 至測試資料集，其中會包含標記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。

ERROR_MANIFEST_TOO_FEW_USABLE_LABELS

錯誤訊息

清單檔案可用標籤太少。

其他資訊

訓練清單檔案可以包含影像層級標籤格式和物件位置格式的 JSON Lines。根據訓練清單檔案中找到的 JSON Lines 類型而定，Amazon Rekognition 自訂標籤可選擇建立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 Lines 篩選出有效的 JSON 記錄。當所選模型類型清單檔案中的標籤數目不足以訓練模型時，即會出現 ERROR_MANIFEST_TOO_FEW_USABLE_LABELS。

訓練偵測影像層級標籤的模型至少需要 1 個標籤。訓練偵測物件位置的模型至少需要 2 個標籤。

修正 ERROR_MANIFEST_TOO_FEW_USABLE_LABELS (主控台)

1. 檢查清單檔案摘要中的 `use_case` 欄位。
2. 針對和 `use_case` 的值相符的使用案例 (影像層級或物件本地化)，將更多標籤新增至訓練資料集。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_FEW_USABLE_LABELS (JSON Line)

1. 檢查清單檔案摘要中的 `use_case` 欄位。
2. 針對和 `use_case` 的值相符的使用案例 (影像層級或物件本地化)，將更多標籤新增至訓練資料集。如需詳細資訊，請參閱建立清單檔案。

ERROR_INSUFFICIENT_USABLE_LABEL_OVERLAP

錯誤訊息

訓練和測試清單檔案之間的可用標籤重疊小於 {}%。

其他資訊

訓練清單檔案可以包含影像層級標籤格式和物件位置格式的 JSON Lines。根據訓練清單檔案中找到的格式而定，Amazon Rekognition 自訂標籤可選擇建立偵測影像層級標籤的模型，或是偵測物件位置的模型。Amazon Rekognition 自訂標籤不會對非所選模型格式的 JSON Lines 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 ERROR_INSUFFICIENT_USABLE_LABEL_OVERLAP。

修正 ERROR_INSUFFICIENT_USABLE_LABEL_OVERLAP (主控台)

- 從訓練資料集中移除標籤。或者，新增更多常用標籤至測試資料集。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。

從訓練資料集 (JSON Line) 移除標籤，以修正 ERROR_INSUFFICIENT_USABLE_LABEL_OVERLAP

- 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 Line 包含多個標籤，則僅移除所需標籤的 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清單檔案中的每個 JSON Line 執行此動作。

  用於偵測物件位置的資料集 — 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，請參閱資訊清單檔案中的物件當地語系化。

新增通用標籤至測試資料集 (JSON Line) 以修正錯誤 ERROR_INSUFFICIENT_USABLE_LABEL_OVERLAP

- 新增 JSON Lines 至測試資料集，其中會包含標記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。

ERROR_FAILED_IMAGES_S3_COPY

錯誤訊息

無法從 S3 儲存貯體複製影像。

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。

To fix ERROR_FAILED_IMAGES_S3_COPY

1. 檢查影像的權限。
2. 如果您使用的是 AWS KMS，請檢查儲存貯體政策。如需詳細資訊，請參閱解密使用 加密的檔案 AWS Key Management Service。

清單檔案的終端錯誤太多。

有太多 JSON Lines 具有終端內容錯誤。

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，請參閱終端清單檔案內容錯誤。

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

非終端 JSON Line 驗證錯誤

本主題會列出 Amazon Rekognition 自訂標籤在訓練期間回報的非終端 JSON Line 驗證錯誤。這些錯誤會在訓練和測試驗證清單檔案中回報。如需詳細資訊，請參閱了解培訓和測試驗證結果清單檔案。更新訓練或測試清單檔案中的 JSON Line ，即可修正非終端 JSON Line 錯誤。您也可以從清單檔案中移除 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，您可能會發現重新建立清單檔案更容易。驗證錯誤通常會發生在手動建立的清單檔案中。如需詳細資訊，請參閱建立清單檔案。如需修正驗證錯誤的相關資訊，請參閱 修正訓練錯誤。部分錯誤可透過使用 Amazon Rekognition 自訂標籤主控台加以修正。

ERROR_MISSING_SOURCE_REF

錯誤訊息

缺少 source-ref 金鑰。

其他資訊

JSON Line source-ref 欄位會提供影像的 Amazon S3 位置。缺少 source-ref 金鑰或拼錯時，即會發生此錯誤。此錯誤通常會發生在手動建立的清單檔案中。如需詳細資訊，請參閱建立清單檔案。

修正 **ERROR_MISSING_SOURCE_REF**

1. 檢查 source-ref 金鑰是否存在並且拼寫正確。完整的 source-ref 索引鍵和值會類似下列內容。是 "source-ref": "s3://bucket/path/image"。
2. 更新或 JSON Line 中的 source-ref 金鑰。或者，從清單檔案中移除 JSON Line。

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

ERROR_INVALID_SOURCE_REF_FORMAT

錯誤訊息

source-ref 值的格式無效。

其他資訊

JSON Line 中存在 source-ref 金鑰，但 Amazon S3 路徑的結構描述不正確。例如，路徑https://....代替S3://....。在手動建立的清單檔案中，通常會發生錯誤 ERROR_INVALID_SOURCE_REF_FORMAT。如需詳細資訊，請參閱建立清單檔案。

修正 **ERROR_INVALID_SOURCE_REF_FORMAT**

1. 檢查結構描述為 "source-ref": "s3://bucket/path/image"。例如 "source-ref": "s3://custom-labels-console-us-east-1-1111111111/images/000000242287.jpg"。
2. 更新或移除清單檔案中的 JSON Line。

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

ERROR_NO_LABEL_ATTRIBUTES

錯誤訊息

找不到標籤屬性。

其他資訊

標籤屬性或標籤屬性 -metadata 金鑰名稱 (或兩者) 無效或缺少。在下列範例中，只要缺少 bounding-box 或 bounding-box-metadata 金鑰 (或兩者)，就會發生 ERROR_NO_LABEL_ATTRIBUTES。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建立的清單檔案中。如需詳細資訊，請參閱建立清單檔案。

修正 **ERROR_NO_LABEL_ATTRIBUTES**

1. 檢查標籤屬性識別碼和標籤屬性識別碼 -metadata 金鑰是否存在以及金鑰名稱是否拼寫正確。
2. 更新或移除清單檔案中的 JSON Line。

您無法使用 Amazon Rekognition 自訂標籤主控台修正 ERROR_NO_LABEL_ATTRIBUTES。

ERROR_INVALID_LABEL_ATTRIBUTE_FORMAT

錯誤訊息

標籤屬性 {} 的格式無效。

其他資訊

標籤屬性金鑰的結構描述缺失或無效。ERROR_INVALID_LABEL_ATTRIBUTE_FORMAT 錯誤通常會發生在手動建立的清單檔案中。如需更多詳細資訊，請參閱 建立清單檔案。

修正 **ERROR_INVALID_LABEL_ATTRIBUTE_FORMAT**

1. 檢查標籤屬性金鑰的 JSON Line 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。標籤屬性金鑰已命名為 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. 更新或移除清單檔案中的 JSON Line。

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

ERROR_INVALID_LABEL_ATTRIBUTE_METADATA_FORMAT

錯誤訊息

標籤屬性中繼資料的格式無效。

其他資訊

標籤屬性中繼資料金鑰的結構描述缺失或無效。ERROR_INVALID_LABEL_ATTRIBUTE_METADATA_FORMAT 錯誤通常會發生在手動建立的清單檔案中。如需詳細資訊，請參閱建立清單檔案。

修正 `ERROR_INVALID_LABEL_ATTRIBUTE_FORMAT`

1. 檢查標籤屬性中繼資料金鑰的 JSON Line 結構描述是否類似下列範例。標籤屬性中繼資料金鑰已命名為 `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. 更新或移除清單檔案中的 JSON Line。

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

ERROR_NO_VALID_LABEL_ATTRIBUTES

錯誤訊息

找不到有效的標籤屬性。

其他資訊

在 JSON Line 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。ERROR_INVALID_LABEL_ATTRIBUTE_FORMAT 錯誤通常會發生在手動建立的清單檔案中。如需更多詳細資訊，請參閱 建立清單檔案。

如果 JSON Line 不是支援的 SageMaker AI 資訊清單格式，Amazon Rekognition 自訂標籤會將 JSON Line 標記為無效，並報告ERROR_NO_VALID_LABEL_ATTRIBUTES錯誤。目前，Amazon Rekognition 自訂標籤支援分類任務和週框方塊格式。如需詳細資訊，請參閱建立清單檔案。

修正 **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. 更新或移除清單檔案中的 JSON Line。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。

ERROR_MISSING_BOUNDING_BOX_CONFIDENCE

錯誤訊息

一或多個週框方塊缺少可信度值。

其他資訊

缺少一或多個物件位置週框方框的可信度金鑰。週框方塊的可信度金鑰位於標籤屬性中繼資料中，如下列範例所示。ERROR_MISSING_BOUNDING_BOX_CONFIDENCE 錯誤通常會發生在手動建立的清單檔案中。如需詳細資訊，請參閱the section called “資訊清單檔案中的物件當地語系化”。

```
"bounding-box-metadata": {
 "objects": [{
  "confidence": 1
 }, {
  "confidence": 1
 }],
```

修正 **ERROR_MISSING_BOUNDING_BOX_CONFIDENCE**

1. 檢查標籤屬性中的 objects 陣列所包含的可信度金鑰數量是否和標籤屬性 annotations 陣列中的物件數量相同。
2. 更新或移除清單檔案中的 JSON Line。

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

ERROR_MISSING_CLASS_MAP_ID

錯誤訊息

類別對應中缺少一或多個類別 ID。

其他資訊

在註釋 (週框方塊) 物件中的 class_id 在標籤屬性中繼資料類別對應 (class-map) 中沒有相符項目。如需詳細資訊，請參閱資訊清單檔案中的物件當地語系化。ERROR_MISSING_CLASS_MAP_ID 錯誤通常會發生在手動建立的清單檔案中。

修正 ERROR_MISSING_CLASS_MAP_ID

1. 檢查每個註釋 (週框方塊) 物件中的 class_id 值是否在 class-map 陣列中具有對應的值，如下列範例所示。annotations 陣列和 class_map 陣列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. 更新或移除清單檔案中的 JSON Line。

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

ERROR_INVALID_JSON_LINE

錯誤訊息

JSON Line 的格式無效。

其他資訊

在 JSON Line 中發現未預期的字元。只包含錯誤資訊的新 JSON Line 會取代 JSON Line。在手動建立的清單檔案中，通常會發生錯誤 ERROR_INVALID_JSON_LINE。如需詳細資訊，請參閱the section called “資訊清單檔案中的物件當地語系化”。

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

修正 **ERROR_INVALID_JSON_LINE**

1. 發生 ERROR_INVALID_JSON_LINE 錯誤時，請開啟清單檔案並瀏覽至 JSON Line。
2. 檢查 JSON Line 不包含無效字元，且必要的 , 或 : 字元並未缺少。
3. 更新或移除清單檔案中的 JSON Line。

ERROR_INVALID_IMAGE

錯誤訊息

影像無效。檢查 S3 路徑和/或影像屬性。

其他資訊

`source-ref` 參考的檔案不是有效的影像。可能的原因包括影像長寬比、影像的大小和影像格式。

如需詳細資訊，請參閱指南和配額。

修正 **ERROR_INVALID_IMAGE**

1. 請檢查以下內容。
   - 影像的長寬比小於 20:1。
   - 影像的大小大於 15 MB
   - 影像為 PNG 或 JPEG 格式。
   - `source-ref` 中影像的路徑正確。
   - 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. 更新或移除清單檔案中的 JSON Line。

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

ERROR_INVALID_IMAGE_DIMENSION

錯誤訊息

影像尺寸不符合允許的尺寸。

其他資訊

`source-ref` 參考的影像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。針對具有週框方塊的影像，回報了 `ERROR_INVALID_IMAGE_DIMENSION`。

如需詳細資訊，請參閱指南和配額。

修正 **`ERROR_INVALID_IMAGE_DIMENSION`** (主控台)

1. 使用 Amazon Rekognition 自訂標籤可處理的尺寸更新 Amazon S3 儲存貯體中的影像。
2. 在 Amazon Rekognition 自訂標籤主控台中，執行以下操作：
   a. 從影像中移除現有的週框方塊。
   b. 重新將週框方塊新增至影像。
   c. 儲存您的變更。

   如需詳細資訊，使用週框方塊標記物件。

修正 **`ERROR_INVALID_IMAGE_DIMENSION`** (SDK)

1. 使用 Amazon Rekognition 自訂標籤可處理的尺寸更新 Amazon S3 儲存貯體中的影像。
2. 呼叫 ListDatasetEntries，以取得影像的現有 JSON Line。對於 `SourceRefContains` 輸入參數，請指定影像的 Amazon S3 位置和檔案名稱。
3. 呼叫 更新資料設定嘗試 並提供影像的 JSON Line。請確保 `source-ref` 的值和 Amazon S3 儲存貯體中的影像位置相符。更新週框方塊註釋，以和更新影像所需的週框方塊尺寸相符。

```
{
 "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"
 }
}
```

ERROR_INVALID_BOUNDING_BOX

錯誤訊息

週框方塊具有離框值。

其他資訊

週框方塊資訊會指定影像框外或包含負值的影像。

如需詳細資訊，請參閱指南和配額。

修正 **ERROR_INVALID_BOUNDING_BOX**

1. 檢查 annotations 陣列中週框方塊的值。

```
"bounding-box": {
 "image_size": [{
  "width": 640,
  "height": 480,
  "depth": 3
 }],
 "annotations": [{
  "class_id": 1,
  "top": 251,
  "left": 399,
  "width": 155,
  "height": 101
 }]
},
```

2. 更新，或者從清單檔案中移除 JSON Line。

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

ERROR_NO_VALID_ANNOTATIONS

錯誤訊息

找不到有效的註釋。

其他資訊

JSON Line 中沒有任何註釋物件包含有效的週框方塊資訊。

修正 **ERROR_NO_VALID_ANNOTATIONS**

1. 更新 annotations 陣列以包含有效的週框方塊物件。此外，請檢查標籤屬性中繼資料中對應的週框方塊資訊 (confidence 和 class_map) 是否正確。如需詳細資訊，請參閱資訊清單檔案中的物件當地語系化。

```
{
 "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. 更新，或者從清單檔案中移除 JSON Line。

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

ERROR_BOUNDING_BOX_TOO_SMALL

錯誤訊息

週框方塊的高度和寬度太小。

其他資訊

週框方塊尺寸 (高度和寬度) 必須大於 1 x 1 像素。

在訓練期間，如果任何影像的尺寸大於 1280 像素 (來源影像不受影響)，Amazon Rekognition 自訂標籤會調整影像的大小。產生的週框方塊高度和寬度必須大於 1 x 1 像素。週框方塊位置存放在物件位置 JSON Line 的 `annotations` 陣列中。如需詳細資訊，請參閱資訊清單檔案中的物件當地語系化

```
"bounding-box": {
 "image_size": [{
  "width": 640,
  "height": 480,
  "depth": 3
 }],
 "annotations": [{
  "class_id": 1,
  "top": 251,
  "left": 399,
  "width": 155,
  "height": 101
 }]
},
```

錯誤資訊會新增至註釋物件。

修正 ERROR_BOUNDING_BOX_TOO_SMALL

- 選擇下列其中一個選項。
  - 增加太小的週框方塊的大小。
  - 移除太小的週框方塊。如需有關移除週框方塊的資訊，請參閱 ERROR_TOO_MANY_BOUNDING_BOXES。
  - 從清單檔案中移除影像 (JSON Line)。

ERROR_TOO_MANY_BOUNDING_BOXES

錯誤訊息

週框方塊超過允許的最大值。

其他資訊

週框方塊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週框方塊，也可以從 JSON Line 中移除它們。

修正 `ERROR_TOO_MANY_BOUNDING_BOXES` (主控台)。

1. 決定要移除的週框方塊。
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. 選擇開始使用。
5. 在左側導覽窗格中，選擇包含您要使用之資料集的專案。
6. 在資料集區段中，選擇要使用的資料集。
7. 在資料集圖庫頁面中，選擇開始標記，以進入標籤模式。
8. 選擇您要從中移除週框方塊的影像。
9. 選擇繪製週框方塊。
10. 在繪圖工具中，選擇您要刪除的週框方塊。
11. 按下鍵盤上的 delete 鍵即可刪除週框方塊。
12. 重複前兩個步驟，直到已刪除足夠的週框方塊為止。
13. 選擇完成
14. 選擇儲存變更，以儲存您所做的變更。
15. 選擇退出，可退出標籤模式。

修正 ERROR_TOO_MANY_BOUNDING_BOXES (JSON Line)。

1. 發生 ERROR_TOO_MANY_BOUNDING_BOXES 錯誤時，請開啟清單檔案並瀏覽至 JSON Line。
2. 針對您要移除的每個週框方塊，移除下列項目。

- 從 annotations 陣列中移除所需的 annotation 物件。
- 從標籤屬性中繼資料中的 objects 陣列中移除對應的 confidence 物件。
- 如果其他週框方塊不再使用，請從 class-map 中移除標籤。

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"
 }
}
```

WARNING_UNANNOTATED_RECORD

警告訊息

記錄未加註釋。

其他資訊

使用 Amazon Rekognition 自訂標籤主控台新增至資料集的影像並未標記。影像的 JSON Line 未用於訓練。

```
{
    "source-ref": "s3://bucket/images/IMG_1186.png",
    "warnings": [
        {
            "code": "WARNING_UNANNOTATED_RECORD",
            "message": "Record is unannotated."
        }
    ]
}
```

修正 WARNING_UNANNOTATED_RECORD

- 使用 Amazon Rekognition 自訂標籤主控台標記影像。如需說明，請參閱將影像層級標籤指派給影像。

WARNING_NO_ANNOTATIONS

警告訊息

未提供任何註釋。

其他資訊

目標定位格式的 JSON Line 未包含任何週框方塊資訊，儘管是由人 (human-annotated = yes) 加上註釋。JSON Line 是有效的，但不用於訓練。如需詳細資訊，請參閱[了解培訓和測試驗證結果清單檔案]。

```
{
    "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."
        }
```

```
    ]
}
```

修正 WARNING_NO_ANNOTATIONS

- 選擇下列其中一個選項。
  - 新增週框方塊 (annotations) 資訊至 JSON Line。如需詳細資訊，請參閱資訊清單檔案中的物件當地語系化。
  - 從清單檔案中移除影像 (JSON Line)。

WARNING_NO_ATTRIBUTE_ANNOTATIONS

警告訊息

未提供屬性註釋。

其他資訊

目標定位格式的 JSON Line 未包含任何週框方塊註釋資訊，儘管是由人 (human-annotated = yes) 加上註釋。annotations 陣列不存在或未填入。JSON Line 是有效的，但不用於訓練。如需詳細資訊，請參閱了解培訓和測試驗證結果清單檔案。

```
{
    "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."
        }
    ]
}
```

修正 WARNING_NO_ATTRIBUTE_ANNOTATIONS

- 選擇下列其中一個選項。
  - 新增一或多個週框方塊 annotation 物件至 JSON Line。如需詳細資訊，請參閱資訊清單檔案中的物件當地語系化。
  - 移除週框方塊屬性。
  - 從清單檔案中移除影像 (JSON Line)。如果 JSON Line 中存在其他有效的週框方塊屬性，您可以改為只從 JSON Line 中移除無效的週框方塊屬性。

ERROR_UNSUPPORTED_USE_CASE_TYPE

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."
        }
    ]
}
```

修正 ERROR_UNSUPPORTED_USE_CASE_TYPE

- 請選擇下列其中一個選項：
  - 根據您要建立的模型類型而定，可將 `type`欄位的值變更為 `groundtruth/image-classification` 或 `groundtruth/object-detection`。如需詳細資訊，請參閱建立清單檔案。
  - 從清單檔案中移除影像 (JSON Line)。

ERROR_INVALID_LABEL_NAME_LENGTH

其他資訊

標籤名稱的長度太長。長度上限為 256 個字元。

修正 ERROR_INVALID_LABEL_NAME_LENGTH

- 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。
  - 從清單檔案中移除影像 (JSON Line)。

改善訓練過的 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 自訂標籤會針對每個標籤提供摘要指標和評估指標。如需有關可用指標的詳細資訊，請參閱 用於評估模型的指標。若要使用指標改善模型，請參閱 改善 Amazon Rekognition 自訂標籤模型。

如果對模型的準確性感到滿意，即可開始使用它。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

- 用於評估模型的指標
- 存取評估指標 (主控台)
- 存取 Amazon Rekognition 自訂標籤評估指標 (SDK)
- 改善 Amazon Rekognition 自訂標籤模型

用於評估模型的指標

模型完成訓練後，Amazon Rekognition 自訂標籤會傳回模型測試的指標，您可將其用於評估模型的效能。本主題會說明您可以使用的指標，以及如何了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作為訓練結果摘要以及每個標籤的指標：

- 精確度
- 取回
- F1

我們提供的每個指標都是常用於評估機器學習模型效能的指標。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的每個影像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。可信度分數是量化模型預測確實性的值。

如果自訂標籤的可信度分數超過閾值，則模型輸出會包含此標籤。預測可以透過下列方式進行分類：

- 相符 — Amazon Rekognition 自訂標籤模型會正確地預測出測試影像中存在自訂標籤。也就是說，預測的標籤也會是該影像的「Ground Truth」標籤。例如，當影像中存在足球時，Amazon Rekognition 自訂標籤即會正確地傳回足球標籤。
- 誤報 — Amazon Rekognition 自訂標籤模型會錯誤地預測出測試影像中存在自訂標籤。也就是說，預測的標籤不是影像的 Ground Truth 標籤。例如，Amazon Rekognition 自訂標籤會傳回足球標籤，但該影像的 Ground Truth 中卻沒有足球標籤。
- 漏報 — Amazon Rekognition 自訂標籤模型未預測出影像中存在自訂標籤，但該影像的「Ground Truth」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。
- 不相符 — Amazon Rekognition 自訂標籤模型會正確地預測出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。

主控台會提供對測試資料集中每個影像的相符、不相符和漏報值的存取。如需詳細資訊，請參閱存取評估指標 (主控台)。

這些預測結果會用於計算每個標籤的下列指標，以及整個測試集的彙總。相同的定義適用於模型在週框方塊層級所做的預測，區別在於所有指標都會在每個測試影像中的每個週框方塊 (預測或 Ground Truth) 上面計算。

聯集上的交集 (IoU) 和物件偵測

聯集上的交集 (IoU) 會測量兩個物件週框方塊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 Ground Truth 週框方塊和預測週框方塊的 IoU 至少為 0.5 時，預測的週框方塊即正確。

假設閾值

Amazon Rekognition 自訂標籤會自動計算每個自訂標籤的假設閥值 (0-1)。您無法設定自訂標籤的假設閾值。就每個標籤的假設閥值的值而言，預測若高於該值，即會被計為相符或誤報。該值會根據您的測試資料集設定。假設閥值會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。

您可以從模型的培訓結果中取得標籤的假定臨界值。如需詳細資訊，請參閱存取評估指標 (主控台)。

對假設閥值的變更通常會用於提升型的精確度和取回率。如需詳細資訊，請參閱改善 Amazon Rekognition 自訂標籤模型。由於您無法針對標籤設定模型的假設閾值，因此可以透過分析具有

DetectCustomLabels 的影像並指定 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 自訂標籤可提供每個標籤的精確度指標，以及整個測試資料集的平均精確度指標。

精確度指在個別標籤的假設閾值下，正確預測 (相符) 佔所有模型預測 (相符和誤報) 的比例。隨著閥值的增加，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，其相符率會高於誤報率。精確度的可能值範圍為 0 到 1，而較高的值即表示較高的精確度。

例如，當模型預測影像中存在足球時，該預測正確的機率有多高？ 假設影像中有 8 顆足球和 5 塊岩石。如果模型的預測為 9 顆足球：8 個正確預測和 1 個誤報，則此範例的精確度為 0.89。但是，如果模型的預測為影像中有 13 顆足球，即為 8 個正確預測和 5 個不正確，則產生的精確度較低。

如需詳細資訊，請參閱精確度和取回率。

取回

Amazon Rekognition 自訂標籤可提供每個標籤的平均取回指標，以及整個測試資料集的平均取回指標。

取回率指在假設閾值之上，正確預測出的測試集標籤比例。這會測量模型在自訂標籤實際出現在測試集影像中時可正確預測自訂標籤的機率。取回率的範圍為 0 到 1。值越高，即表示取回率越高。

例如，如果影像包含 8 顆足球，其中有多少個會正確偵測出來？ 在此範例中，影像有 8 顆足球和 5 塊岩石，如果模型偵測到 5 顆足球，則取回值為 0.62。如果在重新訓練後，新模型偵測到 9 顆足球，包括影像中存在的所有 8 顆球，則取回值為 1.0。

如需詳細資訊，請參閱精確度和取回率。

F1

Amazon Rekognition 自訂標籤會使用 F1 分數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指標，會將所有標籤的精確度和取回率納入考量。(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，模型在取回率和精確度方面的成效就越好。具體而言，分類工作的模型效能通常會以 F1 分數來測量。該分數是在假設閾值下精確度和取回分數的調和平均值。例如，對於精確度為 0.9 且取回率為 1.0 的模型，F1 分數為 0.947。

F1 分數的值如果高，即表示模型在精確度和召回率方面都表現良好。如果模型的成效不佳，例如，具有 0.30 的低精確度以及 1.0 的高取回率，則 F1 分數為 0.46。同樣，如果精確度很高 (0.95) 且取回率低 (0.20)，則 F1 分數為 0.33。在這兩種情況下，F1 分數都很低，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

針對您已訓練的特定模型，並依據您的應用程式而定，您可以使用至 DetectCustomLabels 的 MinConfidence 輸入參數在精確度和取回率之間進行取捨。MinConfidence 值較高時，您通常會取得較高的精確度 (更正確的足球預測)，但取回率會較低 (會錯過較多實際的足球數)。MinConfidence 的值較低時，您會取得較高的取回率 (正確預測出更多實際的足球數)，但精確度較低 (這些預測的錯誤較多)。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，這些指標也會告知您可採取來提升模型效能的步驟。如需詳細資訊，請參閱改善 Amazon Rekognition 自訂標籤模型。

> **Note**
>
> DetectCustomLabels 會傳回範圍從 0 到 100 的預測值，這些預測值會對應 0 到 1 的指標範圍。

存取評估指標 (主控台)

在測試期間，會根據測試資料集評估模型的效能。測試資料集中的標籤會被視為「Ground Truth」，因為它們代表了實際影像所代表的內容。在測試期間，模型會使用測試資料集進行預測。預測的標籤會和 Ground Truth 標籤進行比較，並可在主控台評估頁面中取得結果。

Amazon Rekognition 自訂標籤主控台會顯示整個模型的摘要指標，以及個別標籤的指標。主控台中可用的指標包括精確度取回、F1 分數、可信度和可信度閾值。如需詳細資訊，請參閱改善訓練過的 Amazon Rekognition 自訂標籤模型。

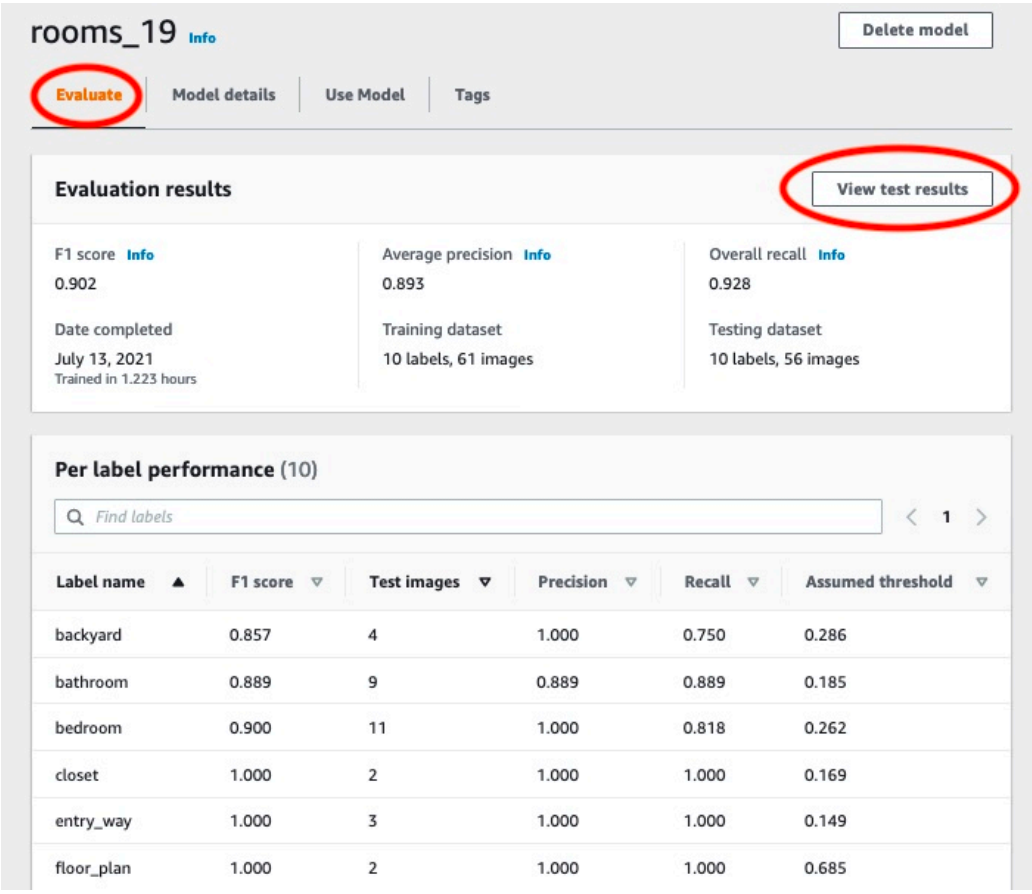
您可以使用主控台以專注於個別指標。例如，若要調查標籤的精確度問題，您可以依標籤和誤報結果篩選訓練結果。如需詳細資訊，請參閱用於評估模型的指標。

訓練後，訓練資料集會變成唯讀。如果您決定提升模型，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複本來訓練新版本的模型。

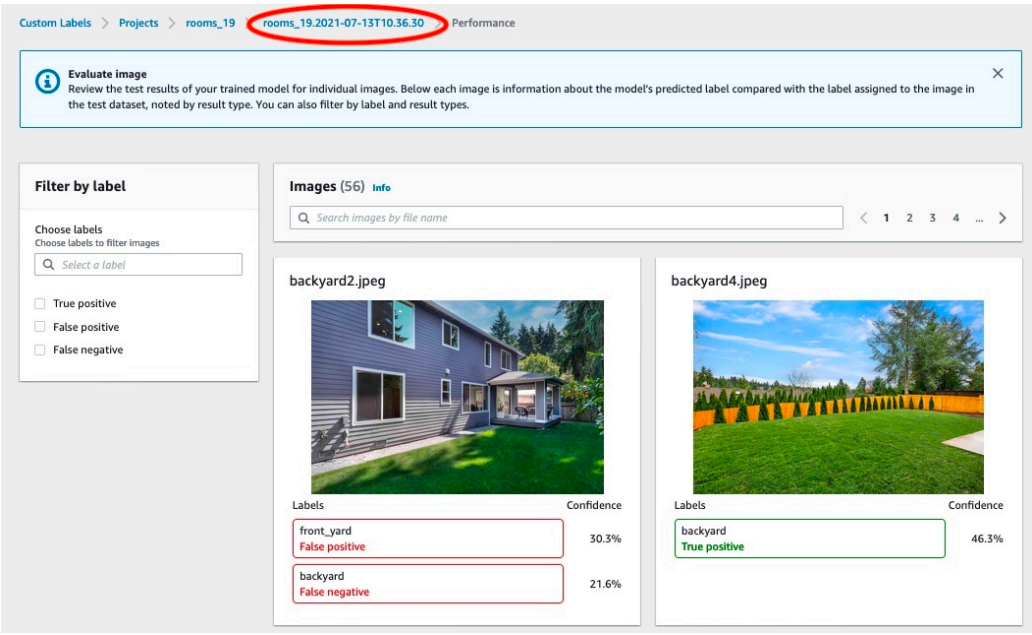
在此步驟中，您會使用主控台來存取主控台中的訓練結果。

存取評估指標 (主控台)

1. 開啟 Amazon Rekognition 主控台：https://console.aws.amazon.com/rekognition/。
2. 選擇使用自訂標籤。
3. 選擇開始使用。
4. 在左側導覽窗格中，選擇專案。
5. 在專案頁面中，選擇包含您要評估的訓練過模型的專案。
6. 在模型中，選擇您要評估的模型。
7. 選擇評估索引標籤，以查看評估結果。如需評估模型的資訊，請參閱 改善訓練過的 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，請參閱用於評估模型的指標。模型評估摘要的下列螢幕擷取畫面顯示 6 個標籤的 F1 分數、平均精確度和整體召回，以及測試結果和效能指標。也會提供使用訓練模型的詳細資訊。



9. 檢視測試結果後，請選擇專案名稱，以返回模型頁面。測試結果頁面顯示以後院和前院影像類別訓練之機器學習模型的預測標籤和可信度分數的影像。顯示兩個範例影像。



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)

DescribeProjectVersions 操作會提供對主控台中所提供的指標以外的指標的存取。

與主控台一樣，DescribeProjectVersions 會提供對下列指標的存取，以作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閾值和個別標籤的閾值。

DescribeProjectVersions 也會提供對下列分類和影像偵測 (影像上的物件位置) 指標的存取。

- 用於影像分類的混淆矩陣。如需詳細資訊，請參閱檢視模型的混淆矩陣。
- 用於影像偵測的平均精確度 (mAP)。
- 用於影像偵測的平均取回率 (mAR)。

DescribeProjectVersions 還會提供對相符、誤報、漏報和不相符值的存取。如需詳細資訊，請參閱用於評估模型的指標。

DescribeProjectVersions 會直接傳回彙總 F1 分數指標。其他指標可從 Amazon S3 儲存貯體中存放的 存取模型摘要檔案 和 解譯評估資訊清單快照 檔案存取。如需詳細資訊，請參閱存取摘要檔案和評估清單檔案快照 (SDK)。

主題

- 存取模型摘要檔案
- 解譯評估資訊清單快照
- 存取摘要檔案和評估清單檔案快照 (SDK)
- 檢視模型的混淆矩陣
- 參考：訓練結果摘要檔案

存取模型摘要檔案

摘要檔案會包含模型的整體評估結果資訊，以及每個標籤的指標。指標為精確度、取回率、F1 分數。也會提供模型的閾值。摘要檔案位置可從由 DescribeProjectVersions 傳回的 EvaluationResult 物件存取。如需詳細資訊，請參閱參考：訓練結果摘要檔案。

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
    }
```

```
  ]
}
```

解譯評估資訊清單快照

評估清單檔案快照會包含測試結果的詳細資訊。快照會包括每個預測的可信度評分。它還會包括和影像實際分類相比的預測分類 (相符、不相符、誤報或漏報)。

由於只包含可用於測試和訓練的影像，因此這些檔案會是快照。無法驗證的影像 (例如格式錯誤的影像) 不會包含在清單檔案中。測試快照位置可從由 `DescribeProjectVersions` 傳回的 `TestingDataResult` 物件存取。訓練快照位置可從由 `DescribeProjectVersions` 傳回的 `TrainingDataResult` 物件存取。

快照採用 SageMaker AI Ground Truth 資訊清單輸出格式，並新增欄位以提供其他資訊，例如偵測二進位分類的結果。下列程式碼片段會顯示其他欄位。

```
"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 — 清單檔案快照中的 `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最小閾值比較結果的預測二進制分類。
- is-present-in-ground-truth — 如果模型所做的預測存在於用於訓練的 Ground Truth 資訊中，即為真，否則即為偽。此值的根據並非可信度分數是否超過模型所計算的最小閾值。
- ground-truth-labeling-jobs — 清單檔案行中用於訓練的 Ground Truth 欄位清單。

如需 SageMaker AI Ground Truth 資訊清單格式的相關資訊，請參閱輸出。

以下是測試清單檔案快照的範例，其中會顯示影像分類和物件偵測的指標。

```
// For image classification
{
```

```
  "source-ref": "s3://amzn-s3-demo-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://amzn-s3-demo-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清單檔案快照 (SDK)

若要取得訓練結果，您可呼叫 DescribeProjectVersions。如需範例程式碼，請參閱 描述一個模型 (SDK)。

指標位置會在 DescribeProjectVersions 的 ProjectVersionDescription 回應中傳回。

- EvaluationResult — 摘要檔案的位置。
- TestingDataResult — 用於測試的評估清單檔案快照的位置。

在 EvaluationResult 中傳回 F1 分數和摘要檔案位置。例如：

```
"EvaluationResult": {
                "F1Score": 1.0,
```

```
                "Summary": {
                    "S3Object": {
                        "Bucket": "echo-dot-scans",
                        "Name": "test-output/EvaluationResultSummary-my-echo-dots-
project-v2.json"
                    }
                }
            }
```

評估清單檔案快照會存放在您在 培訓模型 (SDK) 指定的 `--output-config` 輸入參數中指定的位置。

> **Note**
>
> 針對在 `BillableTrainingTimeInSeconds` 中傳回之訓練的時間向您計費 (秒數)。

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，您可以將改進重點集中在模型上。

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 自訂標籤只會建立分類模型的混淆矩陣。分類矩陣可以從 Amazon Rekognition 自訂標籤在模型訓練期間建立的摘要檔案存取。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
- 檢視模型的混淆矩陣

使用混淆矩陣

下表是 Rooms 影像分類範例專案的混淆矩陣。欄標題是分配給測試影像的標籤 (Ground Truth 標籤)。列標題是模型針對測試影像所預測的標籤。每個儲存格都是標籤 (列) 的預測百分比，並應為 Ground

Truth 標籤 (資料欄)。例如，針對浴室所做的預測中有 67% 會正確地標記為浴室。33% 的浴室則會錯誤地標記為廚房。當預測的標籤和 Ground Truth 標籤相符時，高效能模型會具有較高的儲存格值。您可以看到這些會呈現對角線：從第一個到最後一個預測和 Ground Truth 標籤。如果儲存格值為 0，則不會針對該儲存格的預測標籤進行預測，該標籤應為儲存格的 Ground Truth 標籤。

> **Note**
>
> 由於模型不具確定性，因此您從訓練 Rooms 專案所取得的混淆矩陣儲存格值可能會與下表不同。

混淆矩陣可識別要專注的區域。例如，混淆矩陣會顯示模型有 50% 的時間會將衣櫥和臥室混淆。在此情況下，您應該在訓練資料集中新增更多衣櫥和臥室的影像。還要檢查現有的衣櫥和臥室影像是否標記正確。這應該有助於模型將兩個標籤區分得更清楚。若要將更多影像新增至資料集，請參閱 將更多圖像新增至資料集。

雖然混淆矩陣很實用，但請務必考慮其他指標。例如，100% 的預測正確地找到了 floor_plan 標籤，這表示效能極為卓越。但是，測試資料集只有 2 個具有 floor_plan 標籤的影像。它還有 11 個具有 living_space 標籤的影像。這種不平衡的現象也出現在訓練資料集中 (13 個 living_space 影像和 2 個衣櫥影像)。為了取得更準確的評估，請透過新增更多代表性不足的標籤影像 (此範例中為建築平面圖) 來平衡訓練和測試資料集。若要取得每個標籤的測試影像數目，請參閱 存取評估指標 (主控台)。

下表是混淆矩陣範例，比較預測標籤 （ 在 y 軸上 ） 與 Ground Truth 標籤：

預測標籤	後院	浴室	臥室	衣櫥	入口	建築平面圖	前院	廚房	living_space	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%
衣櫥	0%	0%	0%	50%	0%	0%	0%	0%	0%	0%
入口	0%	0%	0%	0%	33%	0%	0%	0%	0%	0%

預測標籤	後院	浴室	臥室	衣櫥	入口	建築平面圖	前院	廚房	living_space	露台
建築平面圖	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%
living_space	0%	0%	18%	0%	67%	0%	0%	12%	91%	33%
露台	0%	0%	0%	0%	0%	0%	0%	0%	0%	67%

檢視模型的混淆矩陣

下列程式碼會使用 DescribeProjects 和 DescribeProjectVersions 操作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。

顯示模型的混淆矩陣 (SDK)

1. 如果您尚未這麼做，請安裝並設定 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. 使用下列程式碼來顯示模型的混淆矩陣。提供下列命令列參數：

   - `project_name` — 您要使用的專案名稱。您可以從 Amazon Rekognition 自訂標籤主控台中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 自訂標籤主控台中的專案詳細資料頁面取得版本名稱。

```
# 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：訓練結果摘要檔案

訓練結果摘要會包含可用於評估模型的指標。摘要檔案也可用於在主控台訓練結果頁面中顯示指標。摘要檔案會在訓練後存放在 Amazon S3 儲存貯體中。若要取得摘要檔案，請致電 `DescribeProjectVersion`。如需範例程式碼，請參閱 存取摘要檔案和評估清單檔案快照 (SDK)。

摘要檔案

下列 JSON 採用摘要檔案的格式。

EvaluationDetails (區段 3)

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`)。還包括用於訓練 (`NumberOfTrainingImages`) 和評估 (`NumberOfTestingImages`) 的影像數目。

AggregatedEvaluationResults (區段 1)

與測試資料集搭配使用時，您可以使用 `AggregatedEvaluationResults` 來評估訓練過模型的整體效能。包含 `Precision`、`Recall` 和 `F1Score` 指標的彙總指標。針對物件偵測 (影像上的物件位置)，會傳回 `AverageRecall` (mAR) 和 `AveragePrecision` (mAP) 指標。針對分類 (影像中的物件類型)，會傳回混淆矩陣指標。

LabelEvaluationResults (區段 2)

您可以使用 `labelEvaluationResults` 來評估個別標籤的效能。標籤會按每個標籤的 F1 分數排序。包含的指標為 `Precision`、`Recall`、`F1Score`、和 `Threshold` (用於分類)。

檔案名稱採下列格式：`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大部分會依據多種因素而定，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需要重複資料和模型改進，才能實現所需的效能。以下是有關如何改進模型的資訊。

資料

一般而言，您可以利用品質較佳的資料來提升模型的品質。使用訓練影像，清楚地顯示物體或場景，並且不會和不必要的物品雜亂地堆放在一起。對於物件周圍的週框方塊，請使用訓練影像，將物件顯示為完全可見並且不會被其他物件遮擋。

請確保您的訓練和測試資料集與您最終將執行推論的影像類型相符。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週框方塊。這些影像會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 將更多圖像新增至資料集。

減少誤報 (精確度更佳)

- 首先，檢查是否增加假設閾值可讓您保持正確的預測，同時減少誤報。在某些時候，由於特定模型的精確度和呼叫之間的取捨，這會減少效益。您無法設定標籤的假設閾值，但是可以透過指定 `MinConfidence` 輸入參數的高值至 `DetectCustomLabels` 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。
- 您可能會看到一個或多個感興趣的自訂標籤 (A) 始終與同類物件 (但不是您感興趣的標籤) (B) 混淆。為了協助您，請將 B 作為物件類標籤新增到訓練資料集 (以及您取得誤報的影像)。實際上，您正在透過新的訓練影像協助模型學習預測 B 而不是 A。若要將更多影像新增至訓練資料集，請參閱 將更多圖像新增至資料集。
- 您可能會發現模型被兩個自訂標籤 (A 和 B) 混淆 — 具有標籤 A 的測試影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。使用資料集圖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，新增更多與此混淆類型相關的訓練影像，將有助於重新訓練的模型更能區分 A 和 B。若要將影像新增至訓練資料集，請參閱 將更多圖像新增至資料集。

減少漏報 (更佳的取回率)

- 針對假設閾值使用較低的值。您無法設定標籤的假設閾值，但是可以透過指定 `MinConfidence` 輸入參數的較低值至 `DetectCustomLabels` 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。
- 使用更佳的範例來模擬物件及在其中顯示物件的影像的變化。
- 將您的標籤分割成兩個更容易學習的課程。例如，與其使用好的 cookie 和壞的 cookie，您可能需要好的 cookie、燒焦的 cookie 和破碎的 cookie 來協助模型更深入地學習每個獨特的概念。

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。您可以使用 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
- 可用區域
- 啟動 Amazon Rekognition 自訂標籤模型
- 停止 Amazon Rekognition 自訂標籤模型
- 報告使用的執行持續時間和推論單元

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，通常比偵測和使用週框方塊定位物件 (物件偵測) 的模型具有更高的 TPS。
- 模型的複雜性。
- 解析度較高的影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 Logs 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	Description
`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更多詳細資訊，請參閱 CloudWatch metrics for Rekognition。

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。對 檢測自訂標籤 的呼叫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 啟動專案版本 操作開始執行 Amazon Rekognition 自訂標籤模型。

> **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。若要檢查模型準備的目前狀態，請檢查專案的詳細資訊頁面或使用描述專案版本。

模型啟動後，您可以使用 檢測自訂標籤 來分析使用模型的圖像。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。您可以從 主控台直接啟動模型，或使用 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. 選擇開始使用。
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 開始使用。
   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)

您可透過呼叫 啟動模型版本 API 並在 `ProjectVersionArn` 輸入參數中傳遞模型的 Amazon Resource Name (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式方法，用於輪詢特定狀態發生。或者，您可以通過呼叫 描述專案版本 來檢查當前狀態。

啟動模型 (SDK)

1. 如果您尚未這麼做，請安裝並設定 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

   請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。
   - (選用) `--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

請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 停止專案版本 操作停止執行 Amazon Rekognition 自訂標籤模型。

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。您可以從 主控台直接停止模型，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. 選擇開始使用。
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，執行以下操作：

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)

您可以透過呼叫 停止專案版本 API 並在 `ProjectVersionArn` 輸入參數中傳遞模型的 Amazon Resource Name (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. 如果您尚未這麼做，請安裝並設定 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。

請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

請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 月之後培訓並啟動模型，則可以使用 `InServiceInferenceUnits` Amazon CloudWatch 指標來確定模型執行了多少時間，以及該時段使用的 推論單元 數量。

> **Note**
>
> 如果您在 AWS 區域中只有一個模型，您也可以在 CloudWatch `StopProjectVersion`中追蹤對 `StartprojectVersion`和 的成功呼叫，以取得模型的執行時間。如果您在 AWS 區域中執行多個模型，則此方法無法運作，因為指標不包含模型的相關資訊。

或者，您可以使用 AWS CloudTrail 來追蹤對 `StartProjectVersion`和 的呼叫 `StopProjectVersion`（在事件歷史記錄的 `requestParameters` 欄位中包含模型 ARN)。CloudTrail 活動限期為 90 天，但您可以在 CloudTrail Lake 中儲存長達 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。

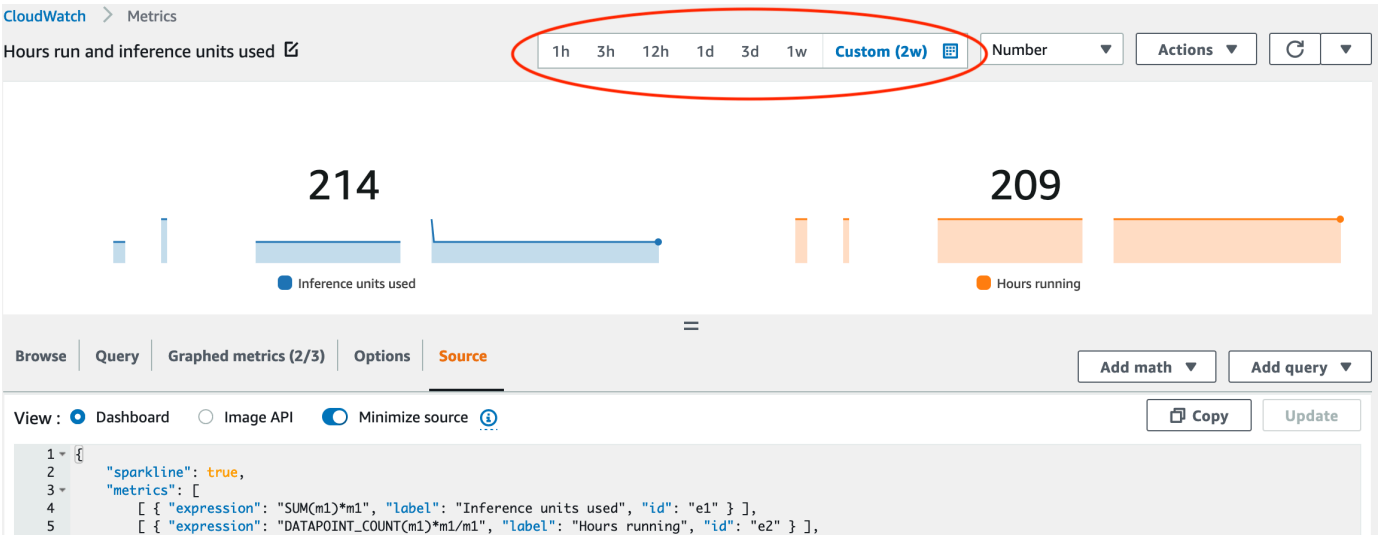
決定模型使用的模型持續時間和推論單元

1. 登入 AWS 管理主控台 ，並在 https：//https://console.aws.amazon.com/cloudwatch/ 開啟 CloudWatch 主控台。
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。變更下列值：
   - `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 AWS 的區域。檢查頁面頂端導覽列中的區域選擇器，確認 CloudWatch 主控台位於相同 AWS 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。以下主控台的螢幕擷取畫面顯示一段時間內使用的推論單位和執行時數，自訂時間設定為 2 週，最高值為 214 個推論單位，執行 209 小時。



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

若要使用經過培訓的 Amazon Rekognition 自訂標籤模型分析圖像，您可以呼叫 偵測自訂標籤 API. DetectCustomLabels 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：

- 影像中找到的物件、場景或概念的標籤。
- 在影像中找到之物件的週框方塊。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。如需更多詳細資訊，請參閱 邊界框。DetectCustomLabels 只有在模型經過培訓以偵測物體位置時，才會傳回邊界框。
- 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 用於評估模型的指標。

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 (精確度更佳)。

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 減少漏報 (更佳的取回率)。

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 培訓模型（主控台）。

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)。偵測自訂標籤的信賴度回應也會以百分比形式傳回。

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 改善訓練過的 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 培訓 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 自訂標籤範例。

如需有關保護 DetectCustomLabels 的資訊，請參閱 保護偵測自訂標籤。

偵測自訂標籤 (API)

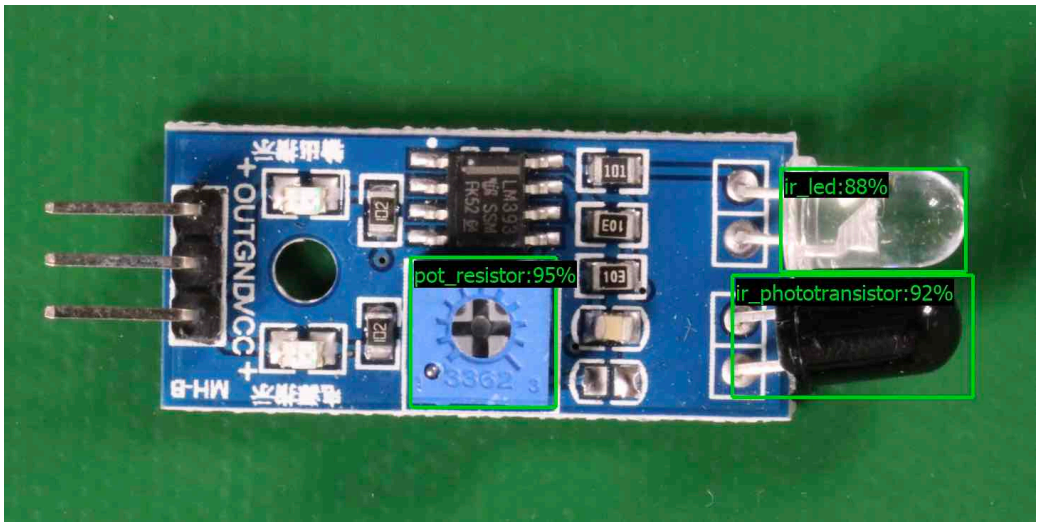
1. 如果您尚未執行：

   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 和 AWS SDKs。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 自訂標籤模型。
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 保護偵測自訂標籤。
4. 將您要分析的圖像上傳到 S3 儲存貯體。

   如需指示說明，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。下列影像包含電路板的週框方塊和標籤，其中包含電位計、紅外線光轉換器和 LED 元件。



AWS CLI

此 AWS CLI 命令會顯示 CLI 操作的 JSON DetectCustomLabels 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 中上傳的輸入影像的檔案名稱。
- projectVersionArn 為您要使用之模型的 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 Pillow 版本 >= 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 – 您要用來分析影像之模型的 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);
        }

    }
}
```

偵測自訂標籤操作請求

在 DetectCustomLabels 的操作中，您以 base64 編碼的位元組數組，或作為儲存在 Amazon S3 儲存貯體中的圖像的形式提供輸入圖像。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,
}
```

偵測自訂標籤操作回應

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

本節提供您用於訓練和管理模型的 Amazon Rekognition 自訂標籤資源概觀。還包括使用 AWS SDK 訓練和使用模型的概觀資訊。

Amazon Rekognition 自訂標籤依賴三種不同的資源來偵測您的自訂標籤：專案、資料集和模型。

- 專案 - 用來將資料集、模型版本和模型評估等其他資源分組在一起。
- 資料集 - 定義用於訓練和測試模型的映像和相關中繼資料。您可以使用 SageMaker AI 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。
- 模型 - 數學模型，透過識別用於訓練模型的影像中的模式，實際預測影像中物件、場景和概念的存在。

主題

- 管理 Amazon Rekognition 自訂標籤專案
- 管理資料集
- 管理 Amazon Rekognition 自訂標籤模型

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案例所建立的模型。專案會管理資料集、模型培訓、模型版本、模型評估，以及專案模型的運作。

主題

- 刪除 Amazon Rekognition 自訂標籤專案
- 描述專案（SDK）
- 使用 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 刪除專案 API 來刪除專案。若要刪除專案，您必須先刪除每個關聯模型。刪除的專案或模型無法恢復。

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)

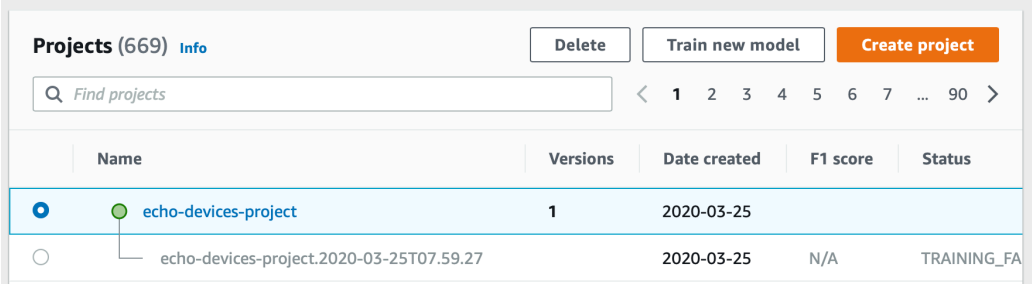
刪除 Amazon Rekognition 自訂標籤專案 (主控台)

您可以從專案頁面刪除專案，也可以從專案的詳細頁面刪除專案。以下流程向您展示了如何使用專案頁面刪除專案。

Amazon Rekognition 自訂標籤主控台會在專案刪除過程中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培訓中，則無法刪除專案。若要停止執行中的模型，請參閱 停止 Amazon Rekognition 自訂標籤模型 (SDK)。如果模型正在培訓中，請等到完成後再刪除專案。

刪除專案（主控台）

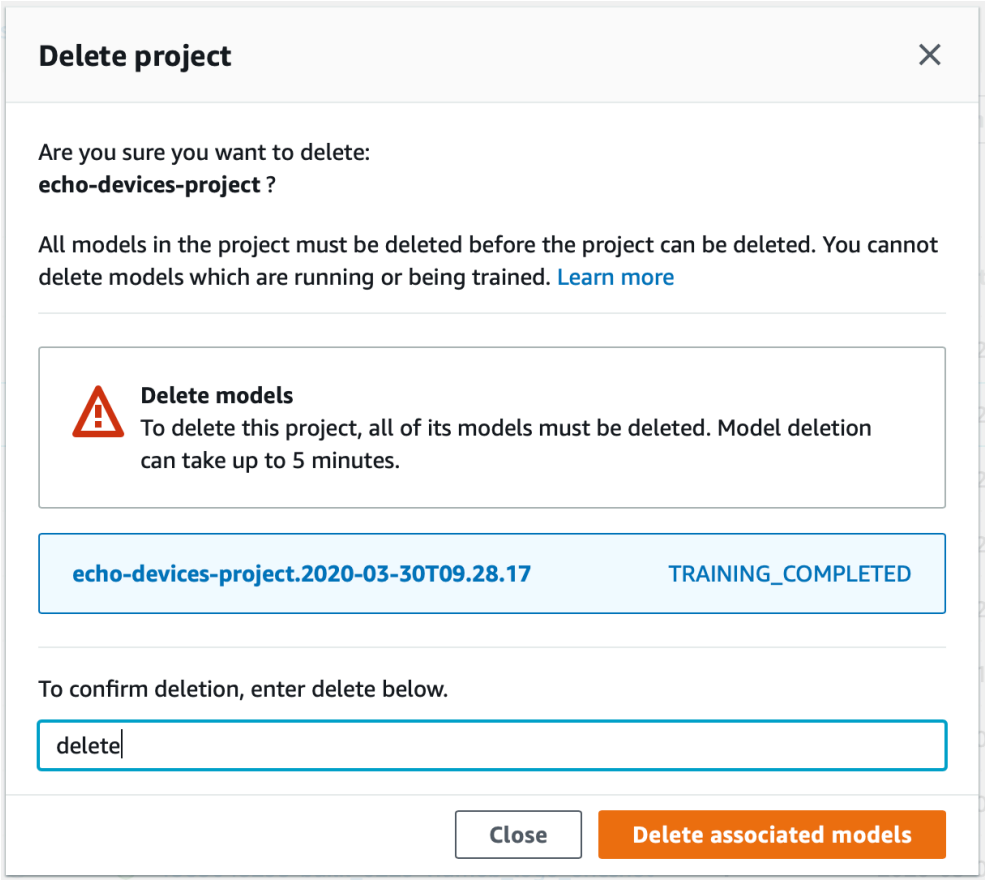
1. 開啟 Amazon Rekognition 主控台：https://console.aws.amazon.com/rekognition/。
2. 選擇使用自訂標籤。
3. 選擇開始使用。
4. 在左側導覽視窗中，選擇 專案。
5. 在 專案 頁面上，選擇要刪除的專案的單選按鈕。專案清單顯示 echo-devices-project，其中 1 個版本建立於 2020-03-25，以及刪除、訓練新模型或建立專案的選項。



6. 在頁面頂部，選擇 刪除。螢幕將會顯示 刪除專案 的對話框。
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. 如果專案具有關聯的模型或資料集：
   a. 輸入 刪除 以確認您要刪除該些模型和資料集。
   b. 選擇 刪除關聯模型 或 刪除關聯資料集 或 刪除關聯資料集和模型，取決於模型是否具有資料集、模型或兩者皆有。刪除模型可能需要一段時間才能完成。

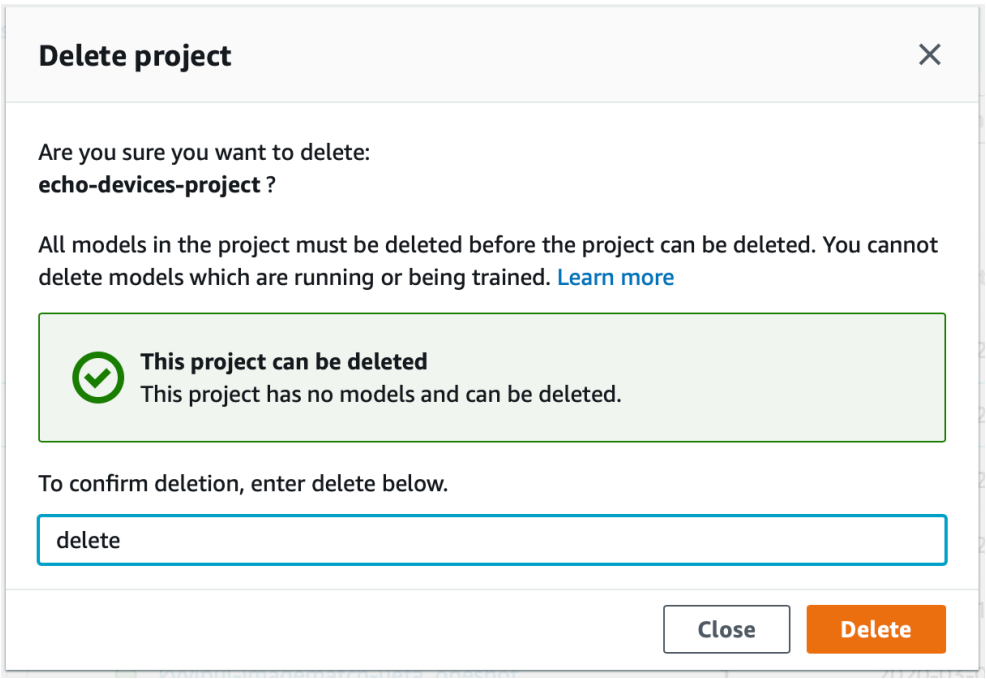
> **Note**
>
> 主控台無法刪除正在培訓或正在運作中的模型。停止所有正在運作中的模型後重試，並等待培訓完成的模型。
> 如果您在刪除模型期間 關閉 了對話框，模型仍會被刪除。稍後，您可以透過重複此過程來刪除該專案。

刪除模型的面板提供刪除相關聯模型的明確指示。



c. 輸入 刪除 以確認您要刪除的專案。

d. 選擇 刪除 以刪除專案。



刪除 Amazon Rekognition 自訂標籤專案 (SDK)

您可以透過呼叫 刪除專案 並提供想要刪除的專案的 Amazon Resource Name (ARN)，以刪除 Amazon Rekognition 自訂標籤專案。若要取得您 AWS 帳戶中專案ARNs，請呼叫 DescribeProjects。回應包括 專案描述 物體的一個數組。專案 ARN 是 `ProjectArn` 欄位。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，必須先刪除專案中的所有模型和資料集。如需更多詳細資訊，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才能刪除。在此期間，該專案的狀態為 `DELETING`。如果對 描述專案 的後續呼叫不包含您刪除的專案，則該專案將被刪除。

刪除專案 (SDK)

1. 如果您尚未這麼做，請安裝並設定 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. 使用下列程式碼刪除專案。

   AWS CLI

   將 `project-arn` 的值變更為要刪除的專案的名稱。

```
aws rekognition delete-project --project-arn project_arn \
  --profile custom-labels-access
```

Python

使用下列程式碼。請提供以下命令列參數：

- project_arn— 您要刪除的專案的 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。請提供以下命令列參數：

- project_arn— 您要刪除的專案的 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);
        }
    }

}
```

描述專案（SDK）

您可以使用 `DescribeProjects` API 來取得專案的相關資訊。

描述專案 (SDK)

1. 如果您尚未這麼做，請安裝並設定 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描述專案。將 `project_name` 替換為您要描述的項目的名稱。若您未指定 `--project-names`，則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

   使用以下程式碼。請提供以下命令列參數：

   - project_name — 您要描述的專案的名稱。若您沒有指定名稱，則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。請提供以下命令列參數：

- project_name— 您要描述的專案的 ARN。若您沒有指定名稱，則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);
        }

    }

}
```

使用 建立專案 AWS CloudFormation

Amazon Rekognition 自訂標籤已與 整合 AWS CloudFormation，此服務可協助您建立和設定 AWS 資源的模型，以減少建立和管理資源和基礎設施的時間。您可以建立範本來描述您想要的所有 AWS 資源，並 CloudFormation 負責為您佈建和設定這些資源。

您可以使用 CloudFormation 來佈建和設定 Amazon Rekognition 自訂標籤專案。

使用 時 CloudFormation，您可以重複使用範本來一致且重複地設定 Amazon Rekognition 自訂標籤專案。只需描述您的專案一次，然後在多個 AWS 帳戶和區域中逐一佈建相同的專案。

Amazon Rekognition 自訂標籤和 CloudFormation 範本

若要預置和配置 Amazon Rekognition 自訂標籤及相關服務的專案，您必須了解 CloudFormation 範本。範本是以 JSON 或 YAML 格式化的文本檔案。這些範本說明您要在 CloudFormation 堆疊中佈建的資源。如果您不熟悉 JSON 或 YAML，您可以使用 CloudFormation 設計工具來協助您開始使用 CloudFormation 範本。如需更多詳細資訊，請參閱 AWS CloudFormation 使用者指南 中的 什麼是 CloudFormation 設計器？。

有關 Amazon Rekognition 自訂標籤專案的參考資訊（包括 JSON 和 YAML 範本的範例），請參閱 Rekognition 資源類型參考。

進一步了解 CloudFormation

若要進一步了解 CloudFormation，請參閱下列資源：

- AWS CloudFormation
- AWS CloudFormation 使用者指南
- CloudFormation API 參考
- AWS CloudFormation 命令列界面使用者指南

管理資料集

資料集包含用於訓練或測試模型的影像和指派的標籤。本節中的主題說明如何使用 Amazon Rekognition 自訂標籤主控台和 AWS SDK 管理資料集。

主題

- 將資料集新增至專案
- 將更多圖像新增至資料集
- 使用現有的資料集建立資料集 (SDK)
- 描述資料集 (SDK)
- 列出資料集條目 (SDK)
- 分配培訓資料集 (SDK)
- 刪除資料集

將資料集新增至專案

您可以將培訓資料集或測試資料集新增至現有專案。如果要取代現有資料集，請先刪除現有的資料集。如需更多詳細資訊，請參閱 刪除資料集。然後，新增新的資料集。

主題

- 將資料集新增至如果您尚未執行此操作，請安裝並設定 專案（主控台）
- 將資料集新增至專案（SDK）

將資料集新增至如果您尚未執行此操作，請安裝並設定 專案（主控台）

您可以使用 Amazon Rekognition 自訂標籤主控台將培訓或測試資料集新增至專案。

將資料集新增至專案

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。畫面將會顯示 Amazon Rekognition 自訂標籤的登入頁面。
3. 在左側導覽視窗中，選擇 專案。畫面將會顯示專案概覽。
4. 選擇您想要新增資料集的專案。
5. 在左側導覽視窗中，專案名稱的下方，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 的頁面。請執行下列操作：
   a. 在 建立資料集 的頁面，輸入圖像來源資訊。如需更多詳細資訊，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. 如果專案擁有現有的資料集 (培訓或測試)，則會顯示專案詳細資料的頁面。請執行下列操作：
   a. 在專案詳細資料頁面上，選擇 動作。
   b. 如果要新增培訓資料集，選擇 建立培訓資料集。
   c. 如果要新增測試資料集，選擇 建立測試資料集。
   d. 在 建立資料集 的頁面中，輸入圖像來源資訊。如需更多詳細資訊，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. 將圖像新增至資料集。如需更多詳細資訊，請參閱 新增更多圖像（主控台）。
9. 在資料集中新增標籤。如需更多詳細資訊，請參閱 新增標籤 (主控台)。

10. 為您的圖像新增標籤。如果您要新增圖像的層階標籤，請參閱 the section called “將影像層級標籤指派給影像”。如果您要新增邊界框，請參閱 使用週框方塊標記物件。如需更多詳細資訊，請參閱規劃資料集。

將資料集新增至專案（SDK）

您可以透過以下方式將培訓或測試資料集新增至現有專案：

- 使用清單檔案建立資料集。如需詳細資訊，請參閱使用 SageMaker AI Ground Truth 資訊清單檔案 (SDK) 建立資料集。
- 建立一個空白的資料集，隨後填入該資料集。以下範例展示如何建立空白的資料集。若要在建立空白的資料集後新增條目，請參閱 將更多圖像新增至資料集。

將資料集新增至專案 (SDK)

1. 如果您尚未這麼做，請安裝並設定 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. 使用以下範例將 JSON 文件新增至資料集。

   CLI

   將 `project_arn` 替換為您想要新增資料集的專案。將 `TRAIN` 替換為 `dataset_type` 以建立培訓資料集，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

   使用以下程式碼建立資料集。提供以下命令列選項：

   - `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（培訓或測試）

   ```
   # 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

使用以下程式碼建立資料集。提供以下命令列選項：

- 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（培訓或測試）

```
/*
   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. 將圖像新增至資料集。如需詳細資訊，請參閱新增更多圖像 (SDK)。

將更多圖像新增至資料集

您可以使用 Amazon Rekognition 自訂標籤主控台或呼叫 `UpdateDatasetEntries` API 為資料集新增更多圖像。

主題

- 新增更多圖像（主控台）
- 新增更多圖像 (SDK)

新增更多圖像（主控台）

當您使用 Amazon Rekognition 自訂標籤主控台時，您可以從本機上傳圖像。圖像將新增至儲存用於建立資料集的影像的 Amazon S3 儲存貯體位置（主控台或外部）。

新增更多圖像至資料集 (主控台)

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。畫面將會顯示 Amazon Rekognition 自訂標籤的登入頁面。
3. 在左側導覽視窗中，選擇 專案。畫面將會顯示專案概覽。
4. 選擇您要使用的專案。
5. 在左側導覽視窗中，專案名稱的下方，選擇 資料集。
6. 選擇 動作，然後選取您要為其新增圖像的資料集。
7. 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。您一次最多可以上傳 30 個影像。
8. 選擇 上傳影像。
9. 選擇 儲存變更。
10. 標記圖像。如需詳細資訊，請參閱標記檔案。

新增更多圖像 (SDK)

UpdateDatasetEntries 會更新 JSON 文件或將 JSON 文件新增至清單檔案。您可以在 GroundTruth 欄位中將 JSON 文件作為 byte64 編碼的資料物體傳遞。如果您使用 AWS 軟體開發套件來呼叫 UpdateDatasetEntries，軟體開發套件會為您編碼資料。每個 JSON 文件包含單一圖像的資訊，例如分配的標籤或邊界框資訊。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 欄位作為索引，以識別您要更新的圖像。如果資料集不包含符合的 source-ref 欄位值，則 JSON 文件將會新增作為新圖像。

新增更多圖像至資料集 (SDK)

1. 如果您尚未這麼做，請安裝並設定 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. 使用以下範例將 JSON 文件新增至資料集。

   CLI

   將 GroundTruth 的值替換為您要使用的 JSON 文件。您需要跳脫 JSON 文件中的任何特殊字符。

```
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。請提供以下命令列參數：

- dataset_arn — 您要更新的資料集的 ARN。
- updates_file — 包含 JSON 文件更新的檔案。

```
# 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

- dataset_arn — 您要更新的資料集的 ARN。
- update_file — 包含 JSON 文件更新的檔案。

```
/*
   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);
        }

    }

}
```

使用現有的資料集建立資料集 (SDK)

以下步驟示範如何使用 建立資料集 的操作從現有的資料集建立資料集。

1. 如果您尚未這麼做，請安裝並設定 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，透過複製另一個資料集來建立資料集。

   AWS CLI

   使用以下程式碼建立資料集。取代以下項目：

   - `project_arn` — 您要為其新增資料集的專案的 ARN。

- dataset_type — 使用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

以下範例會使用現有資料集建立資料集，並會顯示其 ARN。

若要執行該程式，請請提供以下命令列參數：

- 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 — 您要從中建立資料集的該資料集的 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

以下範例會使用現有資料集建立資料集，並會顯示其 ARN。

若要執行該程式，請請提供以下命令列參數：

- 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 — 您要從中建立資料集的該資料集的 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 來獲取有關資料集的資訊。

描述資料集 (SDK)

1. 如果您尚未這麼做，請安裝並設定 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描述資料集。

   AWS CLI

   將 dataset-arn 的值變更為您要描述的資料集的 ARN。

   ```
   aws rekognition describe-dataset --dataset-arn dataset_arn \
     --profile custom-labels-access
   ```

   Python

   使用以下程式碼。請請提供以下命令列參數：

   - dataset_arn — 您要描述的資料集的 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

- dataset_arn — 您要描述的資料集的 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.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);
        }

    }

}
```

列出資料集條目 (SDK)

您可以使用 `ListDatasetEntries` API 列出資料集中每個圖像的 JSON 文件。如需詳細資訊，請參閱建立清單檔案。

列出資料集條目 (SDK)

1. 如果您尚未這麼做，請安裝並設定 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列出資料集中的條目

AWS CLI

將 dataset-arn 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
```

若要只列出有錯誤的 JSON 文件，請指定 has-errors。

```
aws rekognition list-dataset-entries --dataset-arn dataset_arn \
  --has-errors \
  --profile custom-labels-access
```

Python

使用以下程式碼。請請提供以下命令列參數：

- dataset_arn — 您要列出的資料集的 ARN。
- show_errors_only — 如果您只想查看錯誤，請選擇 true 。否則請選擇 false 。

```
# 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

使用以下程式碼。請請提供以下命令列參數：

- dataset_arn — 您要列出的資料集的 ARN。
- show_errors_only — 如果您只想查看錯誤，請選擇 `true` 。否則請選擇 `false` 。

```
/*
   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);
        }

    }

}
```

分配培訓資料集 (SDK)

Amazon Rekognition 自訂標籤需要培訓資料集和測試資料集來培訓您的模型。

如果您使用的是 API，則可以使用 分配培訓資料集 的API 將 20% 的培訓資料集分配到空白的測試資料集中。如果您只有一個可用的清單檔案，則分配培訓資料集可能會很有用。使用單一清單檔案建立培訓資料集。然後建立一個空白的測試資料集並使用 `DistributeDatasetEntries` 以填充測試資料集。

> **Note**
>
> 如果您使用 Amazon Rekognition 自訂標籤主控台並從只有單一資料集的專案開始，Amazon Rekognition 自訂標籤會在培訓期間分割（分配）培訓資料集以建立測試資料集。20% 的培訓資料集條目會移至測試資料集。

分配培訓資料集 (SDK)

1. 如果您尚未這麼做，請安裝並設定 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. 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. 建立您的培訓資料集。如需資料集的更多詳細資訊，請參閱 建立培訓和測試資料集。
4. 建立空白的測試資料集。
5. 使用以下範例程式碼將 20% 的培訓資料集條目指派至測試資料集。您可以透過呼叫 描述專案 來取得專案資料集的 Amazon Resource Names (ARN)。如需範例程式碼，請參閱 描述專案（SDK）。

   AWS CLI

   將 `training_dataset-arn` 和 `test_dataset_arn` 的值變更為您要使用的資料集的 ARNS。

   ```
   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。請請提供以下命令列參數：

- training_dataset_arn — 您將條目分配至該培訓資料集內的 ARN。
- test_dataset_arn — 您將條目分配至測試資料集的 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

使用以下程式碼。請請提供以下命令列參數：

- training_dataset_arn — 您將條目分配至該培訓資料集內的 ARN。
- test_dataset_arn — 您將條目分配至測試資料集的 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培訓和測試資料集。

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)

使用以下步驟刪除資料集。之後，如果專案還有一個剩餘的資料集（培訓或測試），則會顯示專案的詳細頁面。如果專案沒有剩餘的資料集，則會顯示 建立資料集 的頁面。

如果刪除培訓資料集，則必須先為專案建立新的培訓資料集，然後才能培訓模型。如需詳細資訊，請參閱建立包含影像的訓練和測試資料集。

如果刪除測試資料集，您可以培訓模型而無需建立新的測試資料集。在培訓期間，培訓資料集將會被分割以為專案創建新的測試資料集。分割培訓資料集會減少可用於培訓的圖像數量。為了保持品質，我們建議您在培訓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 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。畫面將會顯示 Amazon Rekognition 自訂標籤的登入頁面。
3. 在左側導覽視窗中，選擇 專案。畫面將會顯示專案概覽。
4. 選擇包含要刪除的資料集的專案。
5. 在左側導覽視窗中，專案名稱的下方，選擇 資料集
6. 選擇 動作
7. 若要刪除培訓資料集，選擇 刪除培訓資料集。
8. 若要刪除測試資料集，選擇 刪除測試資料集。
9. 在 刪除 培訓或測試 資料集 的對話框當中，輸入 刪除 以確認您要刪除資料集。
10. 選擇 刪除 培訓或測試 資料集 以刪除資料集。

刪除 Amazon Rekognition 自訂標籤資料集 (SDK)

您可以透過呼叫 刪除資料集 並提供要刪除的資料集的 Amazon Resource Name (ARN) 來刪除 Amazon Rekognition 自訂標籤資料集。若要取得專案內培訓和測試資料集的 ARNs，請呼叫 描述專案。回應包括 專案描述 物體的一個數組。資料集 ARN (`DatasetArn`) 和資料集類型 (`DatasetType`) 皆位於 `Datasets` 清單當中。

如果要刪除培訓資料集，則需要為專案建立新的培訓資料集，然後才能培訓模型。如果要刪除測試資料集，則需要建立新的測試資料集，然後才能培訓模型。如需詳細資訊，請參閱將資料集新增至專案（SDK）。

刪除資料集 (SDK)

1. 如果您尚未這麼做，請安裝並設定 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. 使用以下程式碼刪除資料集。

   AWS CLI

   將 `dataset-arn` 的值變更為要刪除的資料集的 ARN。

```
aws rekognition delete-dataset --dataset-arn dataset-arn \
  --profile custom-labels-access
```

Python

使用下列程式碼。請請提供以下命令列參數：

- dataset_arn — 您要刪除的資料集的 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

使用以下程式碼。請請提供以下命令列參數：

- dataset_arn — 您要刪除的資料集的 ARN。

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

import 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，可預測新影像中是否存在物件、場景和概念。它會透過在用於訓練模型的影像中尋找模式來達到此目的。本區段會說明如何訓練模型、評估其效能以及進行改進。它還會說明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
- 標記模型
- 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

使用 Amazon Rekognition 自訂標籤主控台或使用 DeleteProjectVersion API，即可刪除模型。如果模型正在執行或訓練，則無法刪除該模型。若要停止執行中的模型，請使用 StopProjectVersion API。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該模型。

刪除的模型無法恢復。

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)

下列程序會說明如何從專案詳細資訊頁面刪除模型。您也可以從模型的詳細資訊頁面刪除模型。

刪除模型 (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. 選擇開始使用。
4. 在左側導覽視窗中，選擇專案。
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訓練，且無法刪除。檢查狀態欄位，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，刪除模型對話方塊隨即顯示。
8. 輸入 delete，以確認刪除。
9. 選擇刪除，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框，模型仍會被刪除。

刪除 Amazon Rekognition 自訂標籤模型 (SDK)

呼叫 DeleteProjectVersion 並提供您要刪除的模型的 Amazon Resource Name (ARN)，即可刪除 Amazon Rekognition 自訂標籤模型。您可以從 Amazon Rekognition 自訂標籤主控台中模型詳細資訊頁面的使用您的模型區段取得模型 ARN。或者，可呼叫 DescribeProjectVersions 並提供下列內容。

- 與模型相關聯之專案 (`ProjectArn`) 的 ARN。
- 模型的版本名稱 (`VersionNames`)。

模型 ARN 是來自 `DescribeProjectVersions` 回應的 ProjectVersionDescription 物件中的 `ProjectVersionArn` 欄位。

如果模型正在執行或訓練，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，並檢查模型的 ProjectVersionDescription 物件之 `Status` 欄位。若要停止執行中的模型，請使用 StopProjectVersion API。如需詳細資訊，請參閱停止 Amazon Rekognition 自訂標籤模型 (SDK)。您必須等到模型完成訓練後才能夠將其刪除。

刪除模型 (SDK)

1. 如果您尚未這麼做，請安裝並設定 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. 使用下列程式碼來刪除模型。

   AWS CLI

   將 `project-version-arn` 的值變更為要刪除之專案的名稱。

   ```
   aws rekognition delete-project-version --project-version-arn model_arn \
     --profile custom-labels-access
   ```

   Python

   請提供以下命令列參數：

   - `project_arn` — 包含要刪除之模型的專案的 ARN。
   - `model_arn` — 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 — 包含要刪除之模型的專案的 ARN。
- model_arn — 要刪除之模型版本的 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);
        }

    }

}
```

標記模型

您可以使用標籤來識別、整理、搜尋和篩選 Amazon Rekognition 模型。每個標籤都是由使用者定義的金鑰和值組成的標籤。例如，為有助於確定模型的計費方式，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。

使用標籤來：

- 使用成本分配標籤追蹤模型的計費。如需詳細資訊，請參閱使用成本分配標籤。
- 使用 Identity and Access Management (IAM) 控制對模型的存取。如需詳細資訊，請參閱使用資源標籤來控制對 AWS 資源的存取權。
- 自動化模型管理。例如，您可以執行自動化啟動或停止指令碼，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 SDKs 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、檢視連接至模型的標籤，以及移除標籤。

新增或移除標籤

此程序會說明如何新增標籤至現有模型或從中移除標籤。您也可以在訓練過模型後將標籤新增到新模型。如需詳細資訊，請參閱培訓 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 Amazon Rekognition 主控台：https://console.aws.amazon.com/rekognition/。
2. 選擇開始使用。
3. 在導覽窗格中，選擇專案。
4. 在專案資源頁面上，選擇包含您要標記的模型的專案。
5. 在導覽窗格中，於您先前選擇的專案下選擇模型。
6. 在模型區段中，選擇您要對其新增標籤的模型。
7. 在模型詳細資訊頁面上，選擇標籤索引標籤。
8. 在標籤區段中，選擇管理標籤。
9. 在管理標籤 頁面上，選擇新增標籤。
10. 輸入金鑰和值。
    a. 在金鑰中，輸入金鑰名稱。
    b. 在值中，輸入值。
11. 若要新增更多標籤，請重複步驟 9 和 10。
12. (選用) 若要移除標籤，請選擇要移除的標籤旁的移除。如果您要移除先前儲存的標籤，則會在您儲存變更時移除該標籤。
13. 請選擇儲存變更，以儲存您所做的變更。

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。

若要檢視連接至專案內所有模型的標籤，您必須使用 AWS SDK。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 Amazon Rekognition 主控台：https://console.aws.amazon.com/rekognition/。
2. 選擇開始使用。
3. 在導覽窗格中，選擇專案。
4. 在專案資源頁面上，選擇包含您要檢視的模型的專案。
5. 在導覽窗格中，於您先前選擇的專案下選擇模型。
6. 在模型區段中，選擇您要檢視其標籤的模型。
7. 在模型詳細資訊頁面上，選擇標籤索引標籤。標籤區段中隨即顯示標籤。

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
- 列出連接至模型的標籤。
- 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

當您使用 CreateProjectVersion 操作建立模型時，您可以將標籤新增至該模型。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
```

如需建立及訓練模型的資訊，請參閱 培訓模型 (SDK)。

將標籤新增到現有模型

若要將一或多個標籤新增到現有模型，請使用 TagResource 操作。指定模型的 Amazon Resource Name (ARN) (ResourceArn) 和您要新增的標籤 (Tags)。下列範例會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
```

呼叫 CreateProjectVersion，即可取得模型的 ARN。

列出模型標籤

若要列出連接至模型的標籤，請使用 ListTagsForResource 操作並指定模型的 ARN (ResourceArn)。回應是連接至指定模型之標籤金鑰和值的對應。

```
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"
    }
}
```

若要查看專案中哪些模型具有特定標籤，請呼叫 DescribeProjectVersions 以取得模型清單。然後在 DescribeProjectVersions 的回應中呼叫每個模型的 ListTagsForResource。檢查 ListTagsForResource 的回應，以查看是否存在所需的標籤。

下面的 Python 3 範例會說明如何搜尋所有專案的特定標籤金鑰和值。輸出包括專案 ARN 和模型 ARN，其中找到一個相符的金鑰。

搜尋標籤值

1. 將下列程式碼儲存到名為 find_tag.py 的檔案。

```
# 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。使用您要尋找的金鑰名稱和金鑰值取代*##*和*#*。

```
python find_tag.py key value
```

從模型刪除標籤

若要從模型移除一或多個標籤，請使用 UntagResource 操作。指定要移除的模型 (ResourceArn) 和標籤金鑰 (Tag-Keys) 的 ARN。

```
aws rekognition untag-resource --resource-arn resource-arn \
  --tag-keys '["key1","key2"]' \
  --profile custom-labels-access
```

或者，您可以指定此格式中的 tag-keys。

```
--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，DescribeProjectVersions 會傳回專案中所有模型版本的描述。

描述一個模型 (SDK)

1. 如果您尚未這麼做，請安裝並設定 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. 使用下列範例程式碼來描述模型的版本。

   AWS CLI

   將 project-arn 的值變更為要描述之專案的 ARN。將 version-name 的值變更為要描述之模型的版本。

```
aws rekognition describe-project-versions --project-arn project_arn \
  --version-names version_name \
  --profile custom-labels-access
```

Python

使用以下程式碼。請提供以下命令列參數：

- project_arn — 您要描述之模型的 ARN。
- model_version — 您要描述之模型的版本。

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

使用以下程式碼。請提供以下命令列參數：

- project_arn — 您要描述之模型的 ARN。
- model_version — 您要描述之模型的版本。

```
/*
   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 CopyProjectVersion 操作從來源 Amazon Rekognition 自訂標籤專案複製 Amazon Rekognition 自訂標籤模型版本到目的地專案。目的地專案可以位於不同的 AWS 帳戶，也可以位於相同的 AWS 帳戶中。典型的案例是從開發 AWS 帳戶將測試的模型複製到生產 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 操作具有下列優點。

- 模型行為一致。模型訓練為非確定性，而且無法保證使用相同資料集訓練的兩個模型會做出相同的預測。使用 CopyProjectVersion 複製模型有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。這可為您節省金錢，因為對於每次成功的模型訓練，您都需要付費。

若要將模型複製到不同的 AWS 帳戶，您必須在目的地帳戶中擁有 Amazon Rekognition 自訂標籤專案 AWS 。如需建立專案的資訊，請參閱 建立專案。請務必在目的地 AWS 帳戶中建立專案。

專案政策是一種資源型政策，可為您要複製的模型版本設定複製權限。當目的地專案與來源專案位於不同的 AWS 帳戶時，您將需要使用專案政策。

在同一帳戶中複製模型版本時，您不需要使用專案政策。但是，如果您想要對這些資源具有更多控制，則可以選擇對帳戶間專案使用專案政策。

您可以呼叫 PutProjectPolicy 操作，將專案政策連接至來源專案。

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，您可以使用用於訓練來源模型的資料集來訓練目的地專案中的模型。如需詳細資訊，請參閱培訓 Amazon Rekognition 自訂標籤模型。

若要將模型從來源專案複製到目的地專案，請執行下列操作：

複製模型

1. 建立專案政策文件。
2. 將專案政策連接至來源專案。
3. 使用 CopyProjectVersion 操作複製模型。

若要從專案中移除專案政策，請呼叫 DeleteProjectPolicy。若要取得連接至專案的專案政策清單，請呼叫 ListProjectPolicies。

主題

- 建立專案政策文件。
- 連接專案政策 (SDK)
- 複製模型 (SDK)
- 列出專案政策 (SDK)
- 刪除專案政策 (SDK)

建立專案政策文件。

Rekognition 自訂標籤會使用資源型政策 (稱為專案政策) 來管理模型版本的複製權限。專案政策是 JSON 格式的文件。

專案政策會允許或拒絕將模型版本從來源專案複製到目的地專案的主體權限。如果目的地專案位於不同的 AWS 帳戶中，則需要專案政策。如果目的地專案與來源專案位於同一 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 AWS 帳戶中特定 IAM 角色的複製許可。

下列範例可讓主體 `arn:aws:iam::111111111111:role/Admin` 複製模型版本 `arn:aws:rekognition:us-east-1:123456789012:project/my_project/version/test_1/1627045542080`。

JSON

```
{
  "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、Resource、Principal、和 Effect 是專案政策文件中的必填欄位。
> 唯一支援的 action 是 rekognition:CopyProjectVersion。
> NotAction、NotResource、和 NotPrincipal 是禁止的欄位，且不得出現在專案政策文件中。

如果您未指定專案政策，則與來源專案相同 AWS 帳戶中的委託人仍然可以複製模型，如果委託人具有身分型政策，例如 AmazonRekognitionCustomLabelsFullAccess，該政策提供呼叫 的許可CopyProjectVersion。

下列程序會建立可與 連接專案政策 (SDK) 中的 Python 範例搭配使用的專案政策文件檔案。如果您使用 put-project-policy AWS CLI 命令，請以 JSON 字串的形式提供專案政策。

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 以拒絕複製權限。
   - 主體 — 您要允許或拒絕存取您在 Resource 中指定的模型版本的主體。例如，您可以為不同的帳戶指定 AWS 帳戶主體。 AWS 我們不會限制您可以使用的主體。如需詳細資訊，請參閱指定主體。
   - 資源 — 您要為其指定複製權限之模型版本的 Amazon Resource Name (ARN)。如果您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/*
2. 將專案政策儲存到您的電腦。
3. 依照 連接專案政策 (SDK) 中的指示，將專案政策連接至來源專案。

連接專案政策 (SDK)

呼叫 PutProjectpolicy 操作，即可將專案政策連接至 Amazon Rekognition 自訂標籤專案。

呼叫您要新增的每個專案政策的 PutProjectPolicy，即可將多個專案政策連接至專案。您最多可以將五個專案政策連接至專案。如果您需要連接更多專案政策，則可以請求增加限制。

當您第一次將唯一專案政策連接至專案時，請勿在 PolicyRevisionId 輸入參數中指定修訂 ID。PutProjectPolicy 的回應是 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。Amazon Rekognition 自訂標籤只會保留專案政策的最新修訂版。如果您嘗試更新或刪除先前的專案政策修訂，就會收到 InvalidPolicyRevisionIdException 錯誤訊息。

若要更新現有的專案政策，請在 PolicyRevisionId 輸入參數中指定專案政策的修訂 ID。呼叫 ListProjectPolicies，即可取得專案中專案政策的修訂 ID。

將專案政策連接至來源專案後，您即可將模型從來源專案複製到目的地專案。如需詳細資訊，請參閱複製模型 (SDK)。

若要從專案中移除專案政策，請呼叫 DeleteProjectPolicy。若要取得連接至專案的專案政策清單，請呼叫 ListProjectPolicies。

將專案政策連接至專案 (SDK)

1. 如果您尚未這麼做，請安裝並設定 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. 建立專案政策文件。
3. 使用下列程式碼將專案政策連接至信任 AWS 帳戶中的專案，其中包含您要複製的模型版本。若要取得專案 ARN，請呼叫 DescribeProjects。若要取得模型版本 ARN，請呼叫 DescribeProjectVersions。

   AWS CLI

   變更下列值：

   - project-arn 至信任 AWS 帳戶中來源專案的 ARN，其中包含您要複製的模型版本。
   - policy-name 至您選擇的政策名稱。
   - principal 至您要允許或拒絕存取在 Model version ARN 中指定的模型版本之主體。
   - project-version-arn 至要複製的模型版本之 ARN。

如果您要更新現有的專案政策，請指定 policy-revision-id 參數並提供所需專案政策的修訂 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

使用以下程式碼。請提供以下命令列參數：

- project_arn — 您要連接專案政策的來源專案的 ARN。
- policy_name — 您選擇的政策名稱。
- project_policy — 包含專案政策文件的檔案。
- policy_revision_id — (選用)。如果您要更新現有的專案政策修訂版，請指定專案政策的修訂 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

使用以下程式碼。請提供以下命令列參數：

- project_arn — 您要連接專案政策的來源專案的 ARN。
- project_policy_name — 您選擇的政策名稱。
- project_policy_document — 包含專案政策文件的檔案。
- project_policy_revision_id — (選用)。如果您要更新現有的專案政策修訂版，請指定專案政策的修訂 ID。

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

package com.example.rekognition;

import java.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 複製模型 (SDK) 的指示複製模型版本。

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的地專案。目的地專案可以位於不同的 AWS 帳戶中，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（或者如果您想要授予 AWS 帳戶中複製的模型版本的特定許可），您必須將專案政策連接至來源專案。如需詳細資訊，請參閱建立專案政策文件。。CopyProjectVersion API 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。

除非您設定適當的許可，否則來源 AWS 帳戶對複製到目的地帳戶的模型沒有所有權。

複製模型 (SDK)

1. 如果您尚未這麼做，請安裝並設定 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. 依照 連接專案政策 (SDK) 中的指示，將專案政策連接至來源專案。
3. 如果您要將模型複製到不同的 AWS 帳戶，請確定目的地帳戶中有一個專案 AWS 。
4. 使用下列程式碼將模型版本複製到目的地專案。

AWS CLI

變更下列值：

- source-project-arn 至包含您要複製的模型版本之來源專案的 ARN。
- source-project-version-arn 至您要複製的模型版本之 ARN。
- destination-project-arn 至您要將模型複製到的目的地專案之 ARN。
- version-name 至目的地專案中模型的版本名稱。
- bucket 至您要將來源模型的訓練結果複製到的 S3 儲存貯體。
- folder 至 bucket 中您要將來源模型的訓練結果複製到的資料夾。
- (選用) kms-key-id 至模型的 AWS Key Management Service 金鑰 ID。
- (選用) key 至您選擇的標籤金鑰。
- (選用) value 至您選擇的標籤值。

```
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

使用以下程式碼。請請提供以下命令列參數：

- 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 — 您要複製之來源 AWS 帳戶中模型版本的 ARN。
- destination_project_arn — 您要將模型複製到的目的地專案之 ARN。
- destination_version_name — 目的地專案中模型的版本名稱。
- training_results — 您要將來源模型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。

- (選用) tag_name 至您選擇的標籤金鑰。
- (選用) tag_value 至您選擇的標籤值。

```
# 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

使用以下程式碼。請請提供以下命令列參數：

- 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 — 您要複製之來源 AWS 帳戶中模型版本的 ARN。
- destination_project_arn — 您要將模型複製到的目的地專案之 ARN。
- destination_version_name — 目的地專案中模型的版本名稱。

- output_bucket — 您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的 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 ListProjectPolicies 操作列出連接至 Amazon Rekognition 自訂標籤專案的專案政策。

列出連接至專案的專案政策 (SDK)

1. 如果您尚未這麼做，請安裝並設定 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. 使用下列程式碼列出專案政策。

   AWS CLI

   將 `project-arn` 變更為您要列出其連接的專案政策之專案的 Amazon Resource Name。

```
aws rekognition list-project-policies \
  --project-arn project-arn \
  --profile custom-labels-access
```

   Python

   使用以下程式碼。請提供以下命令列參數：

   - project_arn — 您要列出其連接的專案政策之專案的 Amazon Resource Name。

   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

使用以下程式碼。請提供以下命令列參數：

- project_arn — 具有您要列出的專案政策之專案的 ARN。

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

package com.example.rekognition;

import java.util.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);
        }

    }
```

```
}
```

刪除專案政策 (SDK)

您可以使用 DeleteProjectPolicy 操作，從 Amazon Rekognition 自訂標籤專案刪除現有專案政策的修訂版。如果您要刪除連接至專案之專案政策的所有修訂版，請使用 ListProjectPolicy 來取得連接至專案的每個專案政策的修訂 ID。然後呼叫每個政策名稱的 `DeletePolicy`。

刪除專案政策的修訂版 (SDK)

1. 如果您尚未這麼做，請安裝並設定 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. 使用下列程式碼刪除專案政策。

   DeletePolicy 需要`ProjectARN`、`PolicyName` 和 `PolicyRevisionId`。此 API 則需要 `ProjectARN` 和 `PolicyName`。`PolicyRevisionId` 為選用，但可以因原子更新的目的而包含在內。

   AWS CLI

   變更下列值：

   - `policy-name` 至要刪除的專案政策的名稱。
   - `policy-revision-id` 至要刪除的專案政策修訂版 ID。
   - `project-arn` 至包含要刪除的專案政策修訂版之專案的 Amazon Resource Name。

   ```
   aws rekognition delete-project-policy \
       --policy-name policy-name \
       --policy-revision-id policy-revision-id \
       --project-arn project-arn \
     --profile custom-labels-access
   ```

   Python

   使用以下程式碼。請提供以下命令列參數：

   - `policy-name` — 要刪除的專案政策的名稱。
   - `project-arn` — 包含要刪除的專案政策的專案的 Amazon Resource Name。

- policy-revision-id — 要刪除的專案政策的修訂 ID。

例如：python delete_project_policy.py *policy_name project_arn policy_revision_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 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

使用以下程式碼。請提供以下命令列參數：

- policy-name — 要刪除的專案政策的名稱。
- project-arn — 包含要刪除的專案政策的專案的 Amazon Resource Name。
- policy-revision-id — 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);
        }

    }

}
```

自訂標籤範例

本節包含範例，說明如何使用 Amazon Rekognition 自訂標籤的功能。

範例	Description
使用模型意見回饋改善模型	說明如何使用人工驗證來改進模型以建立新的培訓資料集。
Amazon Rekognition 自訂標籤演示	展示呼叫 DetectCustomLabels 的結果的使用者介面演示。
在影片中偵測自訂標籤	說明如何使用 DetectCustomLabels 處理從影片中擷取的影格。
使用 AWS Lambda 函數分析影像	說明如何使用 DetectCustomLabels Lambda 函數。
從 CSV 檔案建立清單檔案。	說明如何使用 CSV 檔案建立適合尋找與整個圖像 (分類) 相關的 物體、場景和概念 的清單檔案。

使用模型意見回饋改善模型

模型反饋解決方案使您能夠提供有關模型預測的反饋，並透過人工驗證進行改進。視乎不同案例，只有少數圖像的培訓資料集可以成功使用。可能需要更多註釋的訓練集來建立更準確的模型。使用模型反饋解決方案，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反饋解決方案，請參閱 模型反饋解決方案。

持續模型改善的工作流程如下：

1. 培訓模型的第一個版本（可能使用小型培訓資料集）。
2. 為模型反饋解決方案提供未有註釋的資料集。
3. 模型反饋解決方案會使用當前模型。它會啟動人工驗證來註釋新資料集。
4. 根據人工反饋，模型反饋解決方案會產生一個清單檔案，您可以使用該清單檔案建立新模型。

Amazon Rekognition 自訂標籤演示

Amazon Rekognition 自訂標籤示範展示了一個使用者介面，該介面會使用 偵測自訂標籤 API 分析本機電腦中的圖像。

應用程式會顯示您 AWS 帳戶中 Amazon Rekognition 自訂標籤模型的相關資訊。在您選擇正在執行的模型後，您便可以分析本機電腦中的圖像。如有必要，您可以啟動一個模型。您也可以停止正在運作的模型。該應用程式展示了與其他 AWS 服務（例如 Amazon Cognito、Amazon S3 和 Amazon CloudFront）的整合。

如需更多詳細資訊，請參閱 Amazon Rekognition 自訂標籤演示。

在影片中偵測自訂標籤

以下範例示範如何使用 `DetectCustomLabels` 從影片中擷取影格。該程式碼已使用 mov 和 mp4 格式的視訊檔案進行了測試。

將 `DetectCustomLabels` 配合擷取的影格使用

1. 如果您尚未這麼做，請安裝並設定 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. 請確定您已具備 `rekognition:DetectCustomLabels` 和 `AmazonS3ReadOnlyAccess` 的權限。如需更多詳細資訊，請參閱 步驟 4：設定 AWS CLI 和 AWS SDKs。
3. 使用以下範例程式碼。將此值變更為 `videoFile` 視訊檔案的名稱。將 `projectVersionArn` 的值變更為您的 Amazon Rekognition 自訂標籤模型的 Amazon Resource Name (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()
```

使用 AWS Lambda 函數分析影像

AWS Lambda 是一種運算服務，可讓您執行程式碼，而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圖像，而無需建立伺服器來託管應用程式的程式碼。以下說明展示如何在 Python 中建立呼叫 偵測自訂標籤 的 Lambda 函數。此函數會分析提供的圖像，並傳回圖像中找到的標籤列表。這些指引包括範例 Python 程式碼，展示如何使用 Amazon S3 儲存貯體中的圖像或本機電腦提供的圖像呼叫 Lambda 函數。

主題

- 步驟 1：建立 AWS Lambda 函數 （ 主控台 ）
- 步驟 2：(可選) 建立圖層 (主控台)
- 步驟 3：新增 Python 程式碼 (主控台)
- 步驟 4：嘗試使用您的 Lambda 函數

步驟 1：建立 AWS Lambda 函數 （ 主控台 ）

在此步驟中，您會建立空 AWS 函數和 IAM 執行角色，讓您的函數呼叫 `DetectCustomLabels`操作。它還授予對儲存圖像以供分析的 Amazon S3 儲存貯體的存取權限。您也可以為以下內容指定環境變數：

- 您希望您的 Lambda 函數使用的 Amazon Rekognition 自訂標籤模型。
- 您希望模型可使用的信賴度界限。

稍後，您可以將原始程式碼，和選擇性增加的圖層新增至 Lambda 函數。

建立 AWS Lambda 函數 （ 主控台 ）

1. 登入 AWS 管理主控台 ，並在 https：//https://console.aws.amazon.com/lambda/ 開啟 AWS Lambda 主控台。
2. 選擇建立函數 。如需更多詳細資訊，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 函數的名稱 的值。
   - 針對 執行期，選擇 Python 3.10。
4. 選擇 建立函數 來建立 AWS Lambda 函數。

5. 在函數頁面上，選擇 配置 標籤。
6. 在 環境變數 視窗中，選擇 編輯。
7. 新增以下環境變數。針對每個變數，選擇新增環境變數，然後輸入可變金鑰和值。

金鑰	值
模型_ARN	您希望您的 Lambda 函數使用的模型的 Amazon Resource Name (ARN)。您可以從 Amazon Rekognition 自訂標籤主控台中的 模型詳細頁面的 使用模型 標籤取得模型的 ARN。
信賴度	模型對標籤的預測信賴度的最小值 (0–100)。Lambda 函數不會傳回信賴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 的視窗中的 角色名稱 下，選擇執行角色以在 IAM 主控台中開啟該角色。
10. 在 權限 索引標籤中，依次選擇 新增權限、建立內嵌政策。
11. 選擇 JSON 並將現有政策替換為以下政策。

    JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Action": "rekognition:DetectCustomLabels",
            "Resource": "*",
            "Effect": "Allow",
            "Sid": "DetectCustomLabels"
        }
    ]
}
```

12. 選擇 下一步。
13. 在 政策的詳細資料中，輸入政策的名稱，例如 偵測自訂標籤-存取權。
14. 選擇 建立政策。

15. 如果您要將圖像儲存在 Amazon S3 儲存貯體中進行分析，請重複步驟 10-14。

    a. 針對步驟 11，請使用以下政策。將 *####/#####* 替換為您要分析的圖像的 Amazon S3 儲存貯體和資料夾路徑。

       JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "S3Access",
            "Effect": "Allow",
            "Action": "s3:GetObject",
            "Resource": "arn:aws:s3:::bucket/folder path/*"
        }
    ]
}
```

    b. 針對步驟 13，請選擇不同的政策名稱，例如 S3 儲存貯體-存取權。

步驟 2：(可選) 建立圖層 (主控台)

若要執行此範例，您不需要執行此步驟。`DetectCustomLabels` 操作包含在預設的 Lambda Python 環境中，作為適用於 Python (Boto3) 的 AWS SDK 的一部分。如果 Lambda 函數的其他部分需要不在預設 Lambda Python 環境中的最新 AWS 服務更新，請執行此步驟，將最新的 Boto3 SDK 版本作為 layer 新增至函數。

首先，您需要建立一個包含 Boto3 SDK 的 .zip 檔案存檔。然後，您需要建立一個圖層並將該 .zip 檔案存檔新增至該圖層。如需更多詳細資訊，請參閱 將圖層和 Lambda 函數配合使用。

建立並新增圖層（主控台）

1. 開啟命令提示字元並輸入以下命令。

```
pip install boto3 --target python/.
zip boto3-layer.zip -r python/
```

2. 記下壓縮檔案的名稱 (boto3-layer.zip)。您在此過程的步驟 6 中需要使用它。

3. 在 https：//https://console.aws.amazon.com/lambda/ 開啟 AWS Lambda 主控台。
4. 在導覽視窗中，選擇 圖層。
5. 選擇 建立圖層。
6. 輸入 名稱 和 描述 的值。
7. 選擇 上載 .zip 檔案，然後選擇 上載。
8. 在對話框中，選擇您在此過程的步驟 1 中建立的 .zip 檔案封存 (boto3-layer.zip)。
9. 針對相容的執行期，選擇 Python 3.9。
10. 選擇 建立，以建立圖層。
11. 選擇導覽視窗選單圖示。
12. 在導覽視窗中，選擇函數。
13. 在資源清單中，選擇您在 步驟 1：建立 AWS Lambda 函數（主控台） 中建立的函數。
14. 選擇 程式碼 索引標籤。
15. 在 圖層 部份，選擇 新增圖層。
16. 選擇 自訂圖層。
17. 在自訂圖層中，選擇您在步驟 6 中輸入的圖層名稱。
18. 在 版本中，選擇圖層版本,該版本應為 1。
19. 選擇 新增。

步驟 3：新增 Python 程式碼 (主控台)

在此步驟中，您將使用 Lambda 主控台程式碼編輯器將 Python 程式碼新增至 Lambda 函數。此程式碼會分析 `DetectCustomLabels` 提供的圖像，並傳回圖像中找到的標籤清單。提供的圖像可以儲存於 Amazon S3 儲存貯體中或提供作為 byte64 編碼圖像位元組。

新增 Python 程式碼 (主控台)

1. 如果您不在 Lambda 主控台中，請執行以下操作：
   a. 在 https：//https://console.aws.amazon.com/lambda/ 開啟 AWS Lambda 主控台。
   b. 開啟您在 步驟 1：建立 AWS Lambda 函數（主控台） 中建立的 Lambda 函數。
2. 選擇 程式碼 標籤。
3. 在 程式碼來源中，以下列項目取代 lambda_function.py 中的程式碼：

```
# 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：嘗試使用您的 Lambda 函數

在此步驟中，您將使用電腦上的 Python 程式碼將本機圖像或 Amazon S3 儲存貯體中的圖像傳送到 Lambda 函數。由本機傳送的圖像必須小於 6291456 位元組。如果您的圖像檔較大，請將圖像上傳到 Amazon S3 儲存貯體，並使用圖像的 Amazon S3 路徑呼叫腳本。有關將圖像檔案上傳到 Amazon S3 儲存貯體的資訊，請參閱 上傳物體。

請務必在 AWS 建立 Lambda 函數的相同區域中執行程式碼。您可以在 AWS Lambda 主控台函數詳細資訊頁面的導覽列中檢視 Lambda 函數的區域。

如果 AWS Lambda 函數傳回逾時錯誤，請延長 Lambda 函數的逾時期間。如需詳細資訊，請參閱設定函數逾時 （ 主控台 ）。

如需從程式碼叫用 Lambda 函數的詳細資訊，請參閱叫用 AWS Lambda 函數。

嘗試使用您的 Lambda 函數

1. 請確保您已獲得 `lambda:InvokeFunction`權限。您可以使用以下政策。

   您可以從 Lambda 主控台 的函數概述中取得 Lambda 函數的 ARN。

   若要提供存取權限，請為您的使用者、群組或角色新增權限：

   - 中的使用者和群組 AWS IAM Identity Center：

     建立權限合集。請按照《AWS IAM Identity Center 使用者指南》中的建立權限合集說明進行操作。

   - 透過身分提供者在 IAM 中管理的使用者：

建立聯合身分的角色。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。請按照《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》的新增許可到使用者 (主控台) 中的指示。

2. 安裝和設定適用於 Python 的 AWS SDK。如需詳細資訊，請參閱步驟 4：設定 AWS CLI 和 AWS SDKs。
3. 啟動您在 步驟 1：建立 AWS Lambda 函數 （ 主控台 ） 的步驟 7 中指定的 模型。
4. 將以下程式碼儲存到名為 `client.py` 的檔案。

```
# 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。針對命令列參數，提供 Lambda 函數的名稱和要分析的圖像。您可以提供本機圖像的路徑，或儲存在 Amazon S3 儲存貯體中的圖像的 S3 路徑。例如：

```
python client.py function_name s3://bucket/path/image.jpg
```

如果圖像位於 Amazon S3 儲存貯體中，請確保它與您在 步驟 1：建立 AWS Lambda 函數（主控台）的步驟 15 中指定的儲存貯體相同。

如果成功，輸出的將會是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 1：建立 AWS Lambda 函數（主控台） 的步驟 7 中設定的信賴度值。

6. 如果您已完成 Lambda 函數，且該模型未被其他應用程式使用，請停止該模型。請記住在下次要使用 Lambda 函數時 啟動模型。

安全

您可以保護您的專案、模型以及客戶用於檢測自訂標籤的 `DetectCustomLabels` 操作的管理。

有關保護 Amazon Rekognition 的更多詳細資訊，請參閱 Amazon Rekognition 安全性。

保護 Amazon Rekognition 自訂標籤專案的安全

您可以透過指定以身份為基礎的政策中指定的資源級權限來保護您的 Amazon Rekognition 自訂標籤專案。如需更多詳細資訊，請參閱 以身份為基礎和以資源為基礎的政策。

您可以保護的 Amazon Rekognition 自訂標籤資源包括：

資源	Amazon Resource Name (ARN) 格式
專案	arn:aws:rekognition:*:*:專案/##_##/日期時間
模型	arn:aws:rekognition:*:*:專案/##_##/版本/##/日期時間

以下範例政策顯示如何向下列物件授予身份權限：

- 描述所有專案。
- 建立、啟動、停止和使用特定模型進行推理。
- 建立專案。建立並描述特定模型。
- 拒絕建立特定專案。

JSON

```
{
    "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/*"]
        }
    ]
}
```

保護偵測自訂標籤

用於偵測自訂標籤的身份可能與管理 Amazon Rekognition 自訂標籤模型的身份不同。

您可以透過對身份加入政策來保護該身份於 `DetectCustomLabels` 的存取權限。以下範例僅限制對 `DetectCustomLabels` 和特定模型的存取。此身份無權存取任何其他 Amazon Rekognition 的操作。

JSON

```
{
    "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 受管政策，您可以用來控制對 Amazon Rekognition 自訂標籤的存取。如需更多詳細資訊，請參閱 AWS 受管政策：Amazon Rekognition 自訂標籤的完全存取權限。

Amazon Rekognition 自訂標籤中的指南和配額

以下章節提供使用 Amazon Rekognition 自訂標籤時的指南和配額。

支援地區

如需可使用 Amazon Rekognition 自訂標籤 AWS 的區域清單，請參閱《Amazon Web Services 一般參考》中的 AWS 區域和端點。

配額

以下是 Amazon Rekognition 自訂標籤中的限制清單。如需有關可變更之限制的更多詳細資訊，請參閱 AWS 服務限制。若要變更限制，請參閱 建立案例。

培訓

- 支援的檔案格式為 PNG 和 JPEG 圖像格式。
- 模型版本中培訓資料集的最大數量為 1。
- 資料集清單檔案的大小上限為 1 GB。
- 每件物體、每個場景和概念 (分類) 資料集的獨有標籤的最小數量為 2。
- 每件物體位置（偵測）資料集的獨有標籤的最小數量為 1。
- 每個清單檔案的獨有標籤數量上限為 250。
- 每個標籤的最小圖像數量為 1。
- 每件物體位置 (偵測) 資料集的圖像數目上限為 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。
- 每件物體、每個場景和概念 (分類) 資料集的圖像數目上限為 500,000 張。預設值為 250,000。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。您不能申請提高限額。
- 每張圖像的最大標籤數目為 50。
- 圖像中的最小邊界框數量為 0。
- 圖像中的最大邊界框數量為 50。
- Amazon S3 儲存貯體中圖像檔案的最小影像尺寸為 64 x 64 像素。

- Amazon S3 儲存貯體中圖像檔案的最大影像尺寸為 4096 x 4096 像素。
- Amazon S3 儲存貯體中圖像的最大檔案上限為 15 MB。
- 最大圖像畫面比例為 20:1。

測試

- 模型版本中測試資料集的最大數量為 1。
- 資料集清單檔案的大小上限為 1 GB。
- 每件物體、每個場景和概念 (分類) 資料集的獨有標籤的最小數量為 2。
- 每件物體位置（偵測）資料集的獨有標籤的最小數量為 1。
- 每個資料集的獨有標籤數量上限為 250。
- 每個標籤的最小圖像數量為 0。
- 每個標籤的最大圖像數量為 1000。
- 每件物體位置 (偵測) 資料集的圖像數目上限為 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。
- 每件物體、每個場景和概念 (分類) 資料集的圖像數目上限為 500,000 張。預設值為 250,000。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。您不能申請提高限額。
- 每個清單檔案中每張圖像的最小標籤數量為 0。
- 每個清單檔案中每張圖像的最大標籤數量為 50。
- 每個清單檔案中每張圖像中的最小邊界框數量為 0。
- 每個清單檔案中每張圖像中的最大邊界框數量為 50。
- Amazon S3 儲存貯體中圖像檔案的最小影像尺寸為 64 x 64 像素。
- Amazon S3 儲存貯體中圖像檔案的最大影像尺寸為 4096 x 4096 像素。
- Amazon S3 儲存貯體中圖像的最大檔案上限為 15 MB。
- 支援的檔案格式為 PNG 和 JPEG 圖像格式。
- 最大圖像畫面比例為 20:1。

偵測

- 作為原始位元組傳遞的圖像最大上限為 4 MB。

- Amazon S3 儲存貯體中圖像的最大檔案上限為 15 MB。
- 輸入的圖像檔案 (儲存在 Amazon S3 儲存貯體中或以圖像位元組形式提供) 的最小圖像尺寸為 64 x 64 像素。
- 輸入的圖像檔案 (儲存在 Amazon S3 中或以圖像位元組形式提供) 的最大圖像尺寸為 4096 x 4096 像素。
- 支援的檔案格式為 PNG 和 JPEG 圖像格式。
- 最大圖像畫面比例為 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。
- 目的地當中，並存複製工作的數量上限為 5。

Amazon Rekognition 自訂標籤 API 參考

Amazon Rekognition 自訂標籤 API 會記錄成為 Amazon Rekognition API 參考內容的一部份。這是 Amazon Rekognition 自訂標籤 API 操作的列表，其中包括指向對應 Amazon Rekognition API 參考主題的連結。此外，本文檔中的 API 參考連結可前往相應的 Amazon Rekognition 開發人員指南 API 的參考主題。如需更多使用 API 的詳細資訊，請參閱

本節提供工作流程的概觀，以訓練並使用 Amazon Rekognition 自訂標籤模型搭配 主控台和 AWS SDK。

> **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 主控台和 AWS SDK 為您的專案建立資料集。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。如需更多詳細資訊，請參閱 步驟 6：(可選) 將舊資料集與新專案建立關聯。

主題

- 決定模型類型
- 建立模型
- 改善模型
- 啟動模型
- 分析圖像
- 停止模型

決定模型類型

您首先決定要培訓的模型類型，這取決於您的業務目標。例如，您可以培訓模型在社交媒體貼文中尋找您的標誌、識別商店貨架上的產品或對裝配線上的機器零件進行分類。

Amazon Rekognition 自訂標籤可培訓以下類型的模型：

- 尋找物件、場景和概念
- 尋找物件位置
- 尋找品牌的位置

為了協助您決定要培訓的模型類型，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Amazon Rekognition 自訂標籤入門。

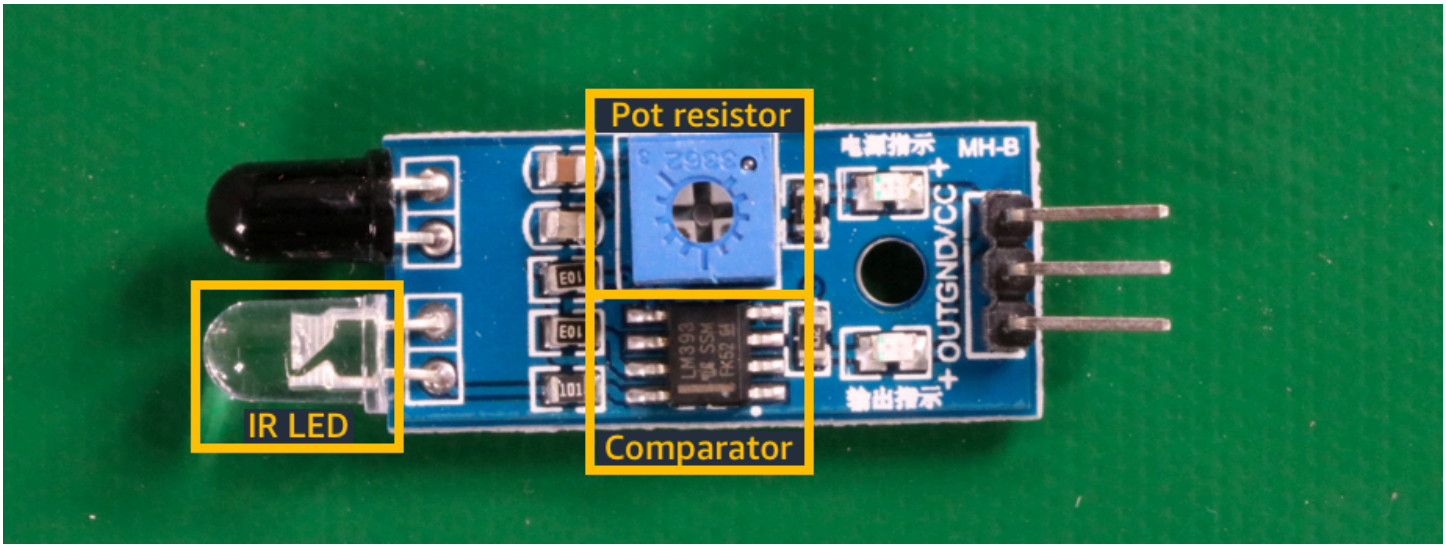
尋找物件、場景和概念

該模型預測與整個圖像相關聯的物件、場景和概念的分類。例如，您可以訓練一個模型來確定圖像是否包含 旅遊景點。如需範例專案，請參閱 Image classification。下列湖泊影像是您可以辨識物件、場景和概念的影像類型範例。

或者，您可以培訓一個將圖像分類為多個類別的模型。例如，上一張圖像可能具有 天空顏色、 反射 或 湖泊 等類別。如需範例專案，請參閱 多標籤影像分類。

尋找物件位置

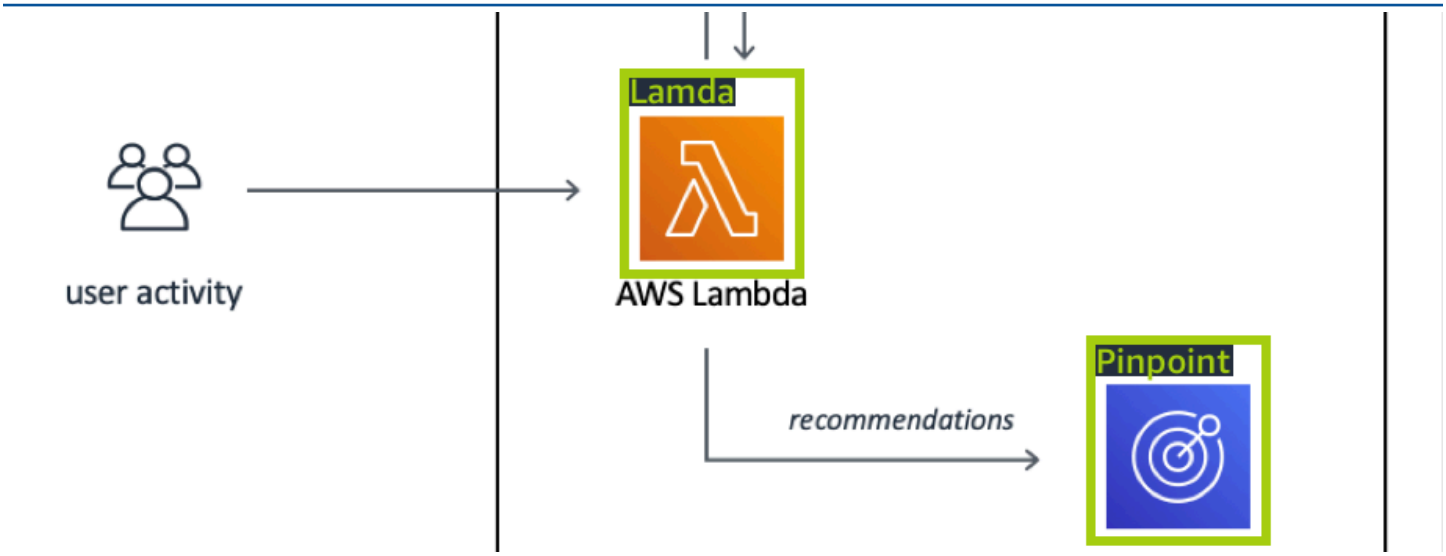
該模型會預測圖像上物件的位置。預測包括物件位置的邊界框資訊，以及用來識別邊界框內物件的標籤。例如，下圖顯示了電路板各個部分，例如 比較器 或 電位器 周圍的邊界框。



物件本地化 的範例專案展示了 Amazon Rekognition 自訂標籤如何使用標籤的邊界框來培訓尋找物件位置的模型。

尋找品牌的位置

Amazon Rekognition 自訂標籤可培訓在圖像上尋找品牌位置（例如商標）的模型。預測包括品牌位置的邊界框資訊，以及用來識別邊界框內物件的標籤。如需範例專案，請參閱 品牌偵測。下圖是模型可以偵測的一些品牌的範例。



建立模型

建立模型的步驟包括建立專案、建立培訓和測試資料集，以及培訓模型。

建立專案

Amazon Rekognition 自訂標籤專案是建立和管理模型所需的一組資源。專案會管理下列項目：

- 資料集 — 用來培訓模型的圖像和圖像標籤。專案備有培訓資料集和測試資料集。

- 模型 — 您培訓此軟體來尋找您的業務獨有的概念、場景和物件。您可以在一個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 建立專案 API 來建立專案。如需詳細資訊，請參閱建立專案。

建立培訓和測試資料集

資料集是描述這些影像的一組影像和標籤。在專案中，您可以建立培訓資料集和測試資料集，Amazon Rekognition 自訂標籤會使用它們來培訓和測試模型。

標籤可識別影像中物件周圍的物件、場景、概念或週框方塊。標籤會指定給整個圖像 (圖像層級)，或將標籤指定給圍繞圖像物件的邊界框。

> **Important**
>
> 如何標記資料集中的圖像，會決定 Amazon Rekognition 自訂標籤所建立的模型類型。例如，若要培訓尋找物件、場景和概念的模型，您可以為培訓和測試資料集中的圖像分配圖像層級標籤。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圖像的建議。如需詳細資訊，請參閱準備影像。

建立培訓和測試資料集 (主控台)

您可以使用單一資料集，或使用個別的培訓和測試資料集來啟動專案。如果您從單一資料集開始，Amazon Rekognition 自訂標籤會在培訓期間分割您的資料集，以便為您的專案建立培訓資料集 (80%) 和測試資料集 (20%)。如果您希望 Amazon Rekognition 自訂標籤決定要用於培訓和測試的圖像，請從單一資料集開始。為了完全控制培訓、測試和效能調整，我們建議您使用個別的培訓和測試資料集來啟動專案。

若要建立專案的資料集，請使用下列其中一種方式匯入圖像：

- 從本機電腦匯入圖像。

- 從 S3 儲存貯體匯入圖像。Amazon Rekognition 自訂標籤可以使用包含圖像的資料夾名稱來標記圖像。
- 匯入 Amazon SageMaker AI Ground Truth 資訊清單檔案。
- 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。

依據您匯入影像的位置而定，您的影像可能沒有標記。例如，從本機電腦匯入的影像即沒有標記。從 Amazon SageMaker AI 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。

若要使用主控台建立培訓和測試資料集，請參閱 建立包含影像的訓練和測試資料集。如需包含建立培訓和測試資料集的教學課程，請參閱 分類映像。

建立培訓和測試資料集 (SDK)

若要建立培訓和測試資料集，請使用 `CreateDataset` API。您可以使用 Amazon SageMaker 格式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。如需更多詳細資訊，請參閱 建立訓練和測試資料集 (SDK)。如有必要，您可以建立自己的清單檔案。如需詳細資訊，請參閱the section called “建立清單檔案”。

培訓模型

使用培訓資料集培訓您的模型。每次培訓模型時，都會建立新版本的模型。在培訓期間，Amazon Rekognition 自訂標籤會測試培訓模型的效能。您可以使用結果來評估和改善模型。培訓需要一段時間才能完成。您只需為成功的模型培訓付費。如需詳細資訊，請參閱培訓 Amazon Rekognition 自訂標籤模型。如果模型培訓失敗，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。

培訓模型（主控台）

若要使用主控台培訓模型，請參閱 培訓模型（主控台）。

培訓模型 (SDK)

您可以呼叫 建立專案版本 來培訓 Amazon Rekognition 自訂標籤模型。如需詳細資訊，請參閱培訓模型 (SDK)。

改善模型

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培訓過的模型。

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) 可讓您了解經過訓練的模型的效能，並決定是否準備好在生產中使用它。如需詳細資訊，請參閱用於評估模型的指標。

評估模型 (主控台)

如要檢視效能指標，請參閱 存取評估指標 (主控台)。

評估模型 (SDK)

要獲取效能指標，請呼叫 描述專案版本 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)。測試結果包括主控台中不可用的指標，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的整體效能的單一值。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，請參閱存取模型摘要檔案。
- 評估清單檔案快照位置 — 快照包含有關測試結果的詳細資料，包括信賴度分級和二進制分類測試的結果，例如誤報。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，請參閱解譯評估資訊清單快照。

改善模型

如果需要改進，您可以新增更多培訓圖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饋，並使用它來改善模型。如需詳細資訊，請參閱使用模型意見回饋改善模型。

改善模型 (主控台)

若要新增影像至資料集，請參閱 將更多圖像新增至資料集。若要加入或變更標籤，請參閱 the section called “標記檔案”。

若要重新培訓模型，請參閱 培訓模型（主控台）。

改善模型 (SDK)

若要將圖像新增至資料集或變更圖像的標籤，請使用 UpdateDatasetEntries API。UpdateDatasetEntries 更新或將 JSON 文件添加至清單檔案。每個 JSON 文件都包含單一圖像的資訊，例如分配的標籤或邊界框資訊。如需詳細資訊，請參閱新增更多圖像 (SDK)。若要檢視資料集中的條目，請使用 ListDatasetEntries API。

若要重新培訓模型，請參閱 培訓模型 (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。您需要根據模型執行時間付費。如需詳細資訊，請參閱執行培訓過的模型。

啟動模型 (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)。

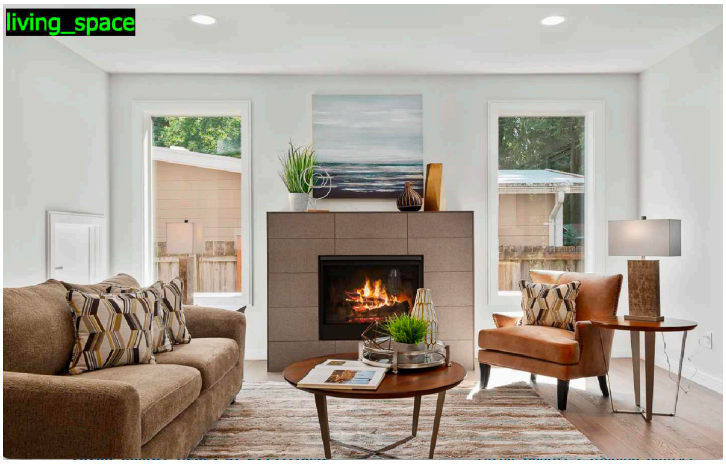
啟動模型

您可以呼叫 啟動模型版本 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)。

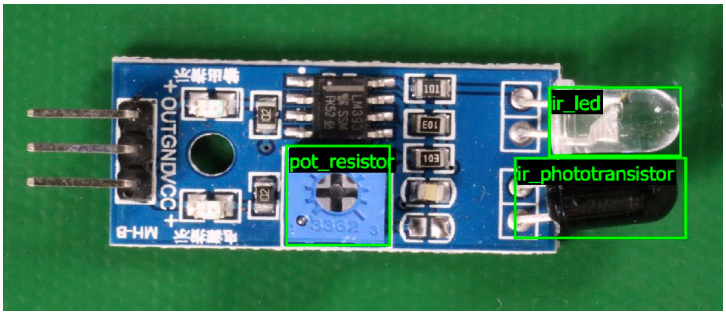
分析圖像

若要使用模型分析圖像，請使用 DetectCustomLabels API。您可以指定本機圖像或存放在 S3 儲存貯體中的圖像。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，則回應會包含在圖像中找到的圖像層級標籤清單。例如，下圖展示使用 Rooms 範例專案找到的圖像層級標籤。



如果模型找到物件位置，則回應會包含在圖像中找到的已標籤的邊界框清單。週框方塊代表物件在影像上的位置。您可以使用邊界框資訊，在物件周圍繪製邊界框。例如，以下圖像展示了使用 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

您需要根據模型的執行時間付費。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。

停止模型（主控台）

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)。

停止模型 (SDK)

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

。

培訓您的模型

專案

- 建立專案 — 建立您的 Amazon Rekognition 自訂標籤專案，是資源（圖像、標籤、模型）和操作（培訓、評估和偵測）的邏輯分組。
- 刪除專案 — 刪除 Amazon Rekognition 自訂標籤專案。
- 描述專案 — 傳回所有 Amazon Rekognition 自訂標籤專案的清單。

專案政策

- PutProjectPolicy：將專案政策連接至信任 AWS 帳戶中的 Amazon Rekognition 自訂標籤專案。
- 列出專案政策 — 傳回附加至專案的專案政策清單。
- 刪除專案政策 — 刪除現有的專案政策。

資料集

- 建立資料集 — 建立 Amazon Rekognition 自訂標籤資料集。
- 刪除資料集 — 刪除 Amazon Rekognition 自訂標籤資料集。
- 描述資料集 — 描述 Amazon Rekognition 自訂標籤資料集。
- 分佈資料集條目 — 將培訓資料集中的條目（圖像）分佈至專案的培訓資料集和測試資料集當中。
- 列出資料集條目 — 傳回 Amazon Rekognition 自訂標籤資料集中的條目（圖像）清單。
- 列出資料集標籤 — 傳回指派給 Amazon Rekognition 自訂標籤資料集的標籤清單。
- 更新資料集條目 — 新增或更新 Amazon Rekognition 自訂標籤資料集中的條目（圖像）。

模型

- 建立專案版本 — 培訓您的 Amazon Rekognition 自訂標籤模型。
- 複製專案版本 — 複製您的 Amazon Rekognition 自訂標籤模型。
- 刪除專案版本 — 刪除 Amazon Rekognition 自訂標籤模型。
- 描述專案版本 — 傳回特定專案中所有 Amazon Rekognition 自訂標籤模型的清單。

標籤

- 標籤資源 — 將一個或多個鍵/值標籤新增至 Amazon Rekognition 自訂標籤模型。
- 取消標籤資源 — 從 Amazon Rekognition 自訂標籤模型中刪除一個或多個標籤。

使用您的模型

- 偵測自訂標籤 — 使用您的自訂標籤模型分析圖像。
- 啟動專案版本 — 啟動您的自訂標籤模型。
- 停止專案版本 — 停止您的自訂標籤模型。

Amazon Rekognition 自訂標籤的文件歷史記錄

下表介紹了 Amazon Rekognition 自訂標籤開發人員指南 每個版本中的重要變更。有關此文件的更新通知，您可以訂閱 RSS 摘要。

- 最新文件更新時間： 2023 年 4 月 19 日

變更	描述	日期
新增了模型持續時間主題	示範如何取得模型所使用的執行時數和推論單元。如需更多詳細資訊，請參閱 報告執行持續時間和使用的推論單元。	2023 年 4 月 19 日
重組的資料集內容	將清單檔案建立內容移至清單檔案。將資料集轉換主題移至將其他資料集格式轉換為清單檔案。	2023 年 2 月 20 日
更新 的 IAM 指引 AWS WAF	更新了指南以符合 IAM 最佳實務。如需更多詳細資訊，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。反之，您可以使用 AWS SDK 來取得並顯示混淆矩陣。如需更多詳細資訊，請參閱 檢視模型的混淆矩陣。	2023 年 1 月 4 日
更新 Lambda 函數範例	Lambda 函數範例現在會示範如從 CSV 檔案建立一個清單檔案何分析從本機檔案或 Amazon S3 儲存貯體傳遞的圖像。如需更多詳細資訊，請參閱 使用 AWS Lambda 函數分析圖像。	2022 年 12 月 2 日

Amazon Rekognition 自訂標籤現在可以複製培訓過的模型	您現在可以將訓練過的模型從一個 AWS 帳戶複製到相同 AWS 區域內的另一個 AWS 帳戶。如需更多詳細資訊，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 自訂標籤現在可以擴展模型使用的推論單元數量。如需更多詳細資訊，請參閱 執行經過培訓的 Amazon Rekognition 自訂標籤模型。	2022 年 8 月 16 日
從 CSV 檔案建立清單檔案	現在，您可以使用從 CSV 檔案讀取圖像分類資訊的原件來簡化清單檔案的建立。如需更多詳細資訊，請參閱 從 CSV 檔案建立清單檔案。	2022 年 2 月 2 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於建立模型的培訓和測試資料集。如需更多詳細資訊，請參閱 了解 Amazon Rekognition 自訂標籤。	2021 年 11 月 1 日
Amazon Rekognition 自訂標籤已與 整合 AWS CloudFormation	您可以使用 CloudFormation 來佈建和設定 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

更新入門體驗	Amazon Rekognition 自訂標籤主控台現在包含了教學影片和專案範例。如需更多詳細資訊，請參閱 Amazon Rekognition 自訂標籤入門。	2021 年 7 月 22 日
更新閾值和使用指標的相關資訊	有關使用 `MinConfidence` 輸入參數至 `DetectCustomLabels` 來設定所需臨界值的資訊。如需更多詳細資訊，請參閱 使用經過培訓的模型分析圖像。	2021 年 6 月 8 日
新增 AWS KMS key 支援	您現在可以使用自己的 KMS 金鑰來加密培訓和測試圖像。如需更多詳細資訊，請參閱 模型培訓。	2021 年 5 月 19 日
新增標記	Amazon Rekognition 自訂標籤現在支援標記功能。您可以使用標記來識別、組織、搜尋和篩選您的 Amazon Rekognition 自訂標籤模型。如需更多詳細資訊，請參閱 標記模型。	2021 年 3 月 25 日
更新設定主題	更新有關如何加密培訓檔案的設定資訊。如需更多詳細資訊，請參閱 步驟 5: (可選) 加密培訓檔案。	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 區域的資訊。如需詳細資訊，請參閱將資料集複製到不同的 AWS 區域。	2021 年 3 月 5 日

新增 Amazon SageMaker AI GroundTruth 多標籤資訊清單轉換主題	如何將 Amazon SageMaker AI 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相關資訊。如需詳細資訊，請參閱轉換多標籤 SageMaker AI Ground Truth 資訊清單檔案。	2021 年 2 月 22 日
新增模型培訓的除錯資訊	現在您可以使用驗證結果的清單檔案來獲取有關模型培訓錯誤的深入除錯資訊。如需更多詳細資訊，請參閱 除錯失敗的模型培訓。	2020 年 10 月 8 日
新增 COCO 轉換資訊和範例	有關如何將 COCO 物體偵測格式資料集轉換為 Amazon Rekognition 自訂標籤清單檔案文檔的資訊。如需更多詳細資訊，請參閱 轉換 COCO 資料集。	2020 年 9 月 2 日
Amazon Rekognition 自訂標籤現在支援單一物體培訓	若要建立尋找單一物體位置的 Amazon Rekognition 自訂標籤模型，您現在可以建立僅需一個標籤的資料集。如需更多詳細資訊，請參閱 繪製邊界框。	2020 年 6 月 25 日
新增專案和模型刪除操作	現在您可以使用主控台和 API 刪除 Amazon Rekognition 自訂標籤專案和模型。如需更多詳細資訊，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

新增了 Java 範例	所新增的 Java 範例，涵蓋了專案建立、模型培訓、模型執行和圖像分析。	2019 年 12 月 13 日
新特徵和指南	這是 Amazon Rekognition 自訂標籤特徵和Amazon Rekognition 自訂標籤開發人員指南的初始版本。	2019 年 12 月 3 日

本文為英文版的機器翻譯版本，如內容有任何歧義或不一致之處，概以英文版為準。