aws

Guía de etiquetas personalizadas

Rekognition

Rekognition: Guía de etiquetas personalizadas

Table of Contents

¿Qué es Etiquetas personalizadas de Amazon Rekognition?

Con Etiquetas personalizadas de Amazon Rekognition, puede identificar los objetos, logotipos y escenas en imágenes que son específicos para las necesidades de su empresa. Por ejemplo, puede encontrar su logotipo en publicaciones de redes sociales, identificar sus productos en los estantes de las tiendas, clasificar piezas de máquinas en una línea de montaje, distinguir plantas en buen estado o infectadas o detectar a personajes animados en vídeos.

Desarrollar un modelo personalizado para analizar imágenes es una tarea importante que requiere tiempo, experiencia y recursos. Además, suele tardar meses. También se necesitan miles o decenas de miles de imágenes etiquetadas manualmente para dar al modelo los datos suficientes para tomar las decisiones pertinentes con precisión. La recopilación de estos datos puede llevar meses y puede exigir que grandes equipos de etiquetadores los preparen para usarlos en el proceso de machine learning.

Etiquetas personalizadas de Amazon Rekognition amplía las anteriores funcionalidades de Amazon Rekognition, que ya se utilizan en decenas de millones de imágenes de muchas categorías. En lugar de miles de imágenes, puede cargar un grupo pequeño de imágenes de entrenamiento (normalmente, unos cientos de imágenes o menos) que sean específicas para cada aplicación. Puede hacerlo mediante la easy-to-use consola. Si las imágenes ya están etiquetadas, Etiquetas personalizadas de Amazon Rekognition podrá empezar a entrenar un modelo en poco tiempo. Si no es así, puedes etiquetar las imágenes directamente en la interfaz de etiquetado o puedes usar Amazon SageMaker AI Ground Truth para etiquetarlas por ti.

Cuando Etiquetas personalizadas de Amazon Rekognition empieza a entrenar a partir de grupo de imágenes, podrá crear un modelo de análisis de imágenes personalizado en tan solo unas horas. En segundo plano, Etiquetas personalizadas de Amazon Rekognition carga e inspecciona automáticamente los datos de entrenamiento, selecciona los algoritmos de machine learning correctos, entrena un modelo y registra las métricas de rendimiento del modelo. Tras esto, podrá usar su modelo personalizado a través de la API de Etiquetas personalizadas de Amazon Rekognition e integrarlo en sus aplicaciones.

Temas

Ventajas principales

Etiquetado de datos mucho más sencillo

La consola de Etiquetas personalizadas de Amazon Rekognition cuenta con una interfaz visual para que etiquetar sus imágenes resulte rápido y sencillo. La interfaz le permite aplicar una etiqueta a toda la imagen. También puede identificar y etiquetar objetos específicos en las imágenes mediante cuadros delimitadores con una click-and-drag interfaz. Como alternativa, si tienes un conjunto de datos grande, puedes usar Amazon SageMaker Ground Truth para etiquetar tus imágenes a escala de manera eficiente.

Aprendizaje automático

No se necesita experiencia en machine learning para crear su modelo personalizado. Etiquetas personalizadas de Amazon Rekognition incluye funciones de machine learning (AutomL) que se encargan del machine learning por usted. Cuando se facilitan las imágenes de entrenamiento, Etiquetas personalizadas de Amazon Rekognition puede cargar e inspeccionar automáticamente los datos, seleccionar los algoritmos de machine learning correctos, entrenar un modelo y registrar las métricas de rendimiento del modelo.

Sistemas sencillos de evaluación, inferencias y respuesta final del modelo

El rendimiento del modelo personalizado se evalúa en el conjunto de pruebas. Para cada imagen del conjunto de prueba, puede ver la side-by-side comparación entre la predicción del modelo y la etiqueta asignada. También puede revisar las métricas de rendimiento de forma detallada, como la precisión, la exhaustividad, las puntuaciones F1 y las puntuaciones de confianza. Puede empezar a utilizar el modelo inmediatamente para el análisis de imágenes, o bien puede iterar y volver a entrenar las nuevas versiones con más imágenes para mejorar el rendimiento. Tras empezar a utilizar el modelo, realice un seguimiento de las predicciones, corrija los errores y utilice los datos de la respuesta final para volver a entrenar las nuevas versiones del modelo y mejorar el rendimiento.

Etiquetas personalizadas de Amazon Rekognition

Amazon Rekognition ofrece dos funciones que puede utilizar para buscar etiquetas (objetos, escenas y conceptos) en las imágenes: Etiquetas personalizadas de Amazon Rekognition y la detección de etiquetas de Amazon Rekognition Image. Utilice la siguiente información para determinar qué característica es mejor utilizar.

Detección de etiquetas de Amazon Rekognition Image

Puede utilizar la característica de detección de etiquetas de Amazon Rekognition Image para identificar, clasificar y buscar etiquetas comunes en imágenes y vídeos, a escala y sin tener que crear un modelo de machine learning. Por ejemplo, puede detectar fácilmente miles de objetos comunes, como automóviles y camiones, tomates, pelotas de baloncesto y balones de fútbol.

Si su aplicación necesita encontrar etiquetas comunes, le recomendamos que utilice la detección de etiquetas de Amazon Rekognition Image, ya que no necesita entrenar un modelo. Para ver la lista de etiquetas que busca la detección de etiquetas de Amazon Rekognition Image, consulte Detección de etiquetas.

Si su aplicación necesita encontrar etiquetas que no ha detectado Amazon Rekognition Image, como componentes de máquinas personalizadas en una línea de montaje, le recomendamos que utilice Etiquetas personalizadas de Amazon Rekognition.

Etiquetas personalizadas de Amazon Rekognition

Puede utilizar Etiquetas personalizadas de Amazon Rekognition para entrenar fácilmente un modelo de machine learning que busque etiquetas (objetos, logotipos, escenas y conceptos) en imágenes que se adapten exclusivamente a las necesidades de su empresa.

Etiquetas personalizadas de Amazon Rekognition puede clasificar las imágenes (predicciones imágenes) o detectar ubicaciones de objetos en una imagen (predicciones objetos/cuadros delimitadores).

Etiquetas personalizadas de Amazon Rekognition ofrece una mayor flexibilidad en los tipos de objetos y escenas que puede detectar. Por ejemplo, puede utilizar la detección de etiquetas de Amazon Rekognition Image para buscar plantas y hojas. Para distinguir entre plantas sanas, marchitas e infectadas, debe utilizar Etiquetas personalizadas de Amazon Rekognition.

En los siguientes ejemplos se ilustra cómo puede usar Etiquetas personalizadas de Amazon Rekognition.

- Identificar logotipos de equipo en camisetas y cascos de jugadores
- Distinguir entre piezas o productos específicos de una máquina en una línea de montaje
- Identificar personajes de dibujos animados en una biblioteca multimedia
- Localizar productos de una marca determinada en los estantes de las tiendas
- Clasificar la calidad de productos agrícolas (como podridos, maduros o crudos)

> **Note**
>
> Etiquetas personalizadas de Amazon Rekognition no sirve para analizar rostros, detectar texto o buscar contenido de imágenes no seguro en las imágenes. Para realizar estas tareas, puede utilizar Amazon Rekognition Image. Para obtener más información, consulte ¿Qué es Amazon Rekognition?

¿Es la primera vez que usa Etiquetas personalizadas de Amazon Rekognition?

Si no lo había usado antes, le recomendamos que lea las siguientes secciones en orden:

1. Configuración de Etiquetas personalizadas de Amazon Rekognition: en esta sección, sabrá cómo crear y ajustar los detalles de la cuenta.
2. Qué es Etiquetas personalizadas de Amazon Rekognition: en esta sección, conocerá el flujo de trabajo para crear un modelo.
3. Introducción a Etiquetas personalizadas de Amazon Rekognition: en esta sección, entrenará un modelo utilizando proyectos de ejemplo creados por Etiquetas personalizadas de Amazon Rekognition.
4. Clasificación de imágenes: en esta sección, aprenderá a entrenar un modelo que clasifique las imágenes con los conjuntos de datos que cree.

Configuración de Etiquetas personalizadas de Amazon Rekognition

En las siguientes instrucciones, aprenderá a configurar la consola de Etiquetas personalizadas de Amazon Rekognition y el SDK.

Tenga en cuenta que puede utilizar la consola de Etiquetas personalizadas de Amazon Rekognition con los siguientes navegadores:

- Chrome: versión 21 o posterior
- Firefox: versión 27 o posterior
- Microsoft Edge: versión 88 o posterior
- Safari: versión 7 o posterior. No es posible usar Safari para dibujar cuadros delimitadores a través de la consola de Etiquetas personalizadas de Amazon Rekognition. Para obtener más información, consulte Etiquetado de objetos con cuadros delimitadores.

Antes de usar Etiquetas personalizadas de Amazon Rekognition por primera vez, complete las siguientes tareas:

Temas

Paso 1: Crea una AWS cuenta

En este paso, crearás una AWS cuenta, crearás un usuario administrativo y aprenderás a conceder acceso programático al AWS SDK.

Temas

- Inscríbase en una Cuenta de AWS
- Creación de un usuario con acceso administrativo
- Acceso programático

Inscríbase en una Cuenta de AWS

Si no tiene uno Cuenta de AWS, complete los siguientes pasos para crearlo.

Para suscribirte a una Cuenta de AWS

1. Abrir https://portal.aws.amazon.com/billing/registro.
2. Siga las instrucciones que se le indiquen.

   Parte del procedimiento de registro implica recibir una llamada telefónica o un mensaje de texto e introducir un código de verificación en el teclado del teléfono.

   Cuando te registras en un Cuenta de AWS, Usuario raíz de la cuenta de AWSse crea un. El usuario raíz tendrá acceso a todos los Servicios de AWS y recursos de esa cuenta. Como práctica recomendada de seguridad, asigne acceso administrativo a un usuario y utilice únicamente el usuario raíz para realizar tareas que requieren acceso de usuario raíz.

AWS te envía un correo electrónico de confirmación una vez finalizado el proceso de registro. En cualquier momento, puede ver la actividad de su cuenta actual y administrarla accediendo a https://aws.amazon.com/y seleccionando Mi cuenta.

Creación de un usuario con acceso administrativo

Después de crear un usuario administrativo Cuenta de AWS, asegúrelo Usuario raíz de la cuenta de AWS AWS IAM Identity Center, habilite y cree un usuario administrativo para no usar el usuario root en las tareas diarias.

Proteja su Usuario raíz de la cuenta de AWS

1. Inicie sesión AWS Management Consolecomo propietario de la cuenta seleccionando el usuario root e introduciendo su dirección de Cuenta de AWS correo electrónico. En la siguiente página, escriba su contraseña.

Para obtener ayuda para iniciar sesión con el usuario raíz, consulte Iniciar sesión como usuario raíz en la Guía del usuario de AWS Sign-In .

2. Active la autenticación multifactor (MFA) para el usuario raíz.

   Para obtener instrucciones, consulte Habilitar un dispositivo MFA virtual para el usuario Cuenta de AWS raíz (consola) en la Guía del usuario de IAM.

Creación de un usuario con acceso administrativo

1. Activar IAM Identity Center.

   Consulte las instrucciones en Activar AWS IAM Identity Center en la Guía del usuario de AWS IAM Identity Center .

2. En IAM Identity Center, conceda acceso administrativo a un usuario.

   Para ver un tutorial sobre su uso Directorio de IAM Identity Center como fuente de identidad, consulte Configurar el acceso de los usuarios con la configuración predeterminada Directorio de IAM Identity Center en la Guía del AWS IAM Identity Center usuario.

Inicio de sesión como usuario con acceso de administrador

- Para iniciar sesión con el usuario de IAM Identity Center, use la URL de inicio de sesión que se envió a la dirección de correo electrónico cuando creó el usuario de IAM Identity Center.

  Para obtener ayuda para iniciar sesión con un usuario del Centro de identidades de IAM, consulte Iniciar sesión en el portal de AWS acceso en la Guía del AWS Sign-In usuario.

Concesión de acceso a usuarios adicionales

1. En IAM Identity Center, cree un conjunto de permisos que siga la práctica recomendada de aplicar permisos de privilegios mínimos.

   Para conocer las instrucciones, consulte Create a permission set en la Guía del usuario de AWS IAM Identity Center .

2. Asigne usuarios a un grupo y, a continuación, asigne el acceso de inicio de sesión único al grupo.

Para conocer las instrucciones, consulte Add groups en la Guía del usuario de AWS IAM Identity Center .

Acceso programático

Los usuarios necesitan acceso programático si quieren interactuar con personas AWS ajenas a. AWS Management Console La forma de conceder el acceso programático depende del tipo de usuario que acceda. AWS

Para conceder acceso programático a los usuarios, elija una de las siguientes opciones.

¿Qué usuario necesita acceso programático?	Para	Mediante
Identidad del personal (Usuarios administrados en el IAM Identity Center)	Usa credenciales temporales para firmar las solicitudes programáticas dirigidas al AWS CLI AWS SDKs, o. AWS APIs	Siga las instrucciones de la interfaz que desea utilizar: • Para ello AWS CLI, consulte Configuración del AWS CLI uso AWS IAM Identity Center en la Guía del AWS Command Line Interface usuario. • Para AWS SDKs ver las herramientas y AWS APIs, consulte la autenticación del Centro de Identidad de IAM en la Guía de referencia de herramientas AWS SDKs y herramientas.
IAM	Utilice credenciales temporales para firmar las solicitudes programáticas dirigidas al AWS CLI AWS SDKs, o. AWS APIs	Siga las instrucciones de Uso de credenciales temporales con AWS recursos de la Guía del usuario de IAM.

¿Qué usuario necesita acceso programático?	Para	Mediante
IAM	(No recomendado) Utilice credenciales de larga duración para firmar las solicitudes programáticas dirigidas al AWS CLI AWS SDKs, o. AWS APIs	Siga las instrucciones de la interfaz que desea utilizar: • Para ello AWS CLI, consulte Autenticación con credencia les de usuario de IAM en la Guía del AWS Command Line Interface usuario. • Para obtener AWS SDKs información sobre las herramientas, consulte Autenticarse con credencia les de larga duración en la Guía de referencia de herramientas AWS SDKs y herramientas. • Para ello AWS APIs, consulte Administrar las claves de acceso para los usuarios de IAM en la Guía del usuario de IAM.

Paso 2: Configurar los permisos de la consola de Etiquetas personalizadas de Amazon Rekognition

Para usar la consola de Amazon Rekognition, debe tener los permisos adecuados. Si desea almacenar sus archivos de entrenamiento en un bucket diferente al bucket de consola, necesitará permisos adicionales.

Temas

- Permiso de acceso a la consola
- Acceso a buckets de Amazon S3 externos

- Asignación de permisos

Permiso de acceso a la consola

Para utilizar la consola Amazon Rekognition Custom Labels, necesita la siguiente política de IAM que cubre Amazon S3 SageMaker , AI Ground Truth y Amazon Rekognition Custom Labels. Para conocer cómo asignar permisos, consulte Asignación de permisos.

```
{
    "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": "*"
        }
    ]
}
```

Acceso a buckets de Amazon S3 externos

Cuando abre por primera vez la consola Amazon Rekognition Custom Labels en una nueva AWS región, Amazon Rekognition Custom Labels crea un depósito (depósito de consola) que se utiliza para almacenar los archivos del proyecto. También puede utilizar su propio bucket de Amazon S3 (bucket externo) para subir las imágenes o el archivo de manifiesto en la consola. Para usar un bucket externo, agregue el siguiente bloque de políticas a la política anterior. Sustituya `amzn-s3-demo-bucket` por el nombre del 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*"
            ]
        }
```

Asignación de permisos

Para dar acceso, agregue permisos a los usuarios, grupos o roles:

- Usuarios y grupos en: AWS IAM Identity Center

  Cree un conjunto de permisos. Siga las instrucciones de Creación de un conjunto de permisos en la Guía del usuario de AWS IAM Identity Center .
- Usuarios gestionados en IAM a través de un proveedor de identidades:

  Cree un rol para la federación de identidades. Siga las instrucciones descritas en Creación de un rol para un proveedor de identidad de terceros (federación) en la Guía del usuario de IAM.
- Usuarios de IAM:
  - Cree un rol que el usuario pueda aceptar. Siga las instrucciones descritas en Creación de un rol para un usuario de IAM en la Guía del usuario de IAM.
  - (No recomendado) Adjunte una política directamente a un usuario o añada un usuario a un grupo de usuarios. Siga las instrucciones descritas en Adición de permisos a un usuario (consola) de la Guía del usuario de IAM.

Paso 3: Crear un bucket de consola

Utilice un proyecto de Etiquetas personalizadas de Amazon Rekognition para crear y administrar sus modelos. Cuando abre por primera vez la consola Amazon Rekognition Custom Labels en una nueva AWS región, Amazon Rekognition Custom Labels crea un bucket de Amazon S3 (bucket de consola) para almacenar sus proyectos. Debe anotar el nombre del bucket de la consola en algún lugar donde pueda consultarlo más adelante, ya que es posible que necesite usar el nombre del bucket en las operaciones del AWS SDK o en las tareas de la consola, como la creación de un conjunto de datos.

El formato del nombre del bucket es `custom-labels-console-<region>-<random value>`. El valor aleatorio garantiza que no haya colisiones entre los nombres de los buckets.

Cómo crear un bucket de consola

1. Asegúrese de que el usuario tenga los permisos correspondientes. Para obtener más información, consulte Permiso de acceso a la consola.
2. Inicie sesión en la consola Amazon Rekognition AWS Management Console y ábrala en. https://console.aws.amazon.com/rekognition/
3. Elija Comenzar.
4. Si es la primera vez que abre la consola en la región de AWS actual, realice lo siguiente en el cuadro de diálogo Configuración inicial:
   a. Anote el nombre del bucket de Amazon S3 que aparece. Necesitará esta información más tarde.
   b. Seleccione Crear bucket de S3 para permitir que Etiquetas personalizadas de Amazon Rekognition cree un bucket de Amazon S3 (bucket de consola) en su nombre.
5. Cierre la ventana del navegador.

Paso 4: Configure y AWS CLIAWS SDKs

Puede utilizar las etiquetas AWS Command Line Interface personalizadas Amazon Rekognition con () y.AWS CLI AWS SDKs Si necesita ejecutar las operaciones de Etiquetas personalizadas de Amazon Rekognition a través del terminal, instale la AWS CLI. Si va a crear una aplicación, descargue el AWS SDK del lenguaje de programación que utilice.

Temas

- Instale los AWS SDK
- Concesión de acceso programático
- Configuración de permisos de SDK
- Llamar a una operación de Etiquetas personalizadas de Amazon Rekognition

Instale los AWS SDK

Siga los pasos para descargar y configurar la AWS SDKs.

Para configurar el AWS CLI y el AWS SDKs

- Descargue e instale el AWS CLIy el AWS SDKs que desee usar. Esta guía proporciona ejemplos AWS CLI de Java y Python. Para obtener información sobre la instalación AWS SDKs, consulte Herramientas para Amazon Web Services.

Concesión de acceso programático

Puede ejecutar los ejemplos de código AWS CLI y los ejemplos de código de esta guía en su ordenador local o en otros AWS entornos, como una instancia de Amazon Elastic Compute Cloud. Para ejecutar los ejemplos, debes conceder acceso a las operaciones del AWS SDK que utilizan los ejemplos.

Temas

- Ejecución del código en su equipo local
- Ejecutar código en AWS entornos

Ejecución del código en su equipo local

Para ejecutar código en un equipo local, te recomendamos que utilices credenciales de corta duración para permitir que un usuario acceda a las operaciones AWS del SDK. Para obtener información específica sobre la ejecución del código AWS CLI y los ejemplos de código en un equipo local, consulteUso de un perfil en su equipo local.

Los usuarios necesitan acceso mediante programación si quieren interactuar con personas AWS ajenas a. AWS Management Console La forma de conceder el acceso programático depende del tipo de usuario que acceda. AWS

Para conceder acceso programático a los usuarios, elija una de las siguientes opciones.

¿Qué usuario necesita acceso programático?	Para	Mediante
Identidad del personal (Usuarios administrados en el IAM Identity Center)	Usa credenciales temporales para firmar las solicitudes programáticas dirigidas al	Siga las instrucciones de la interfaz que desea utilizar: • Para ello AWS CLI, consulte Configuración del AWS

¿Qué usuario necesita acceso programático?	Para	Mediante
	AWS CLI AWS SDKs, o. AWS APIs	CLI uso AWS IAM Identity Center en la Guía del AWS Command Line Interface usuario. • Para AWS SDKs ver las herramientas y AWS APIs, consulte la autenticación del Centro de Identidad de IAM en la Guía de referencia de herramientas AWS SDKs y herramientas.
IAM	Utilice credenciales temporales para firmar las solicitudes programáticas dirigidas al AWS CLI AWS SDKs, o. AWS APIs	Siga las instrucciones de Uso de credenciales temporales con AWS recursos de la Guía del usuario de IAM.

¿Qué usuario necesita acceso programático?	Para	Mediante
IAM	(No recomendado) Utilice credenciales de larga duración para firmar las solicitudes programáticas dirigidas al AWS CLI AWS SDKs, o. AWS APIs	Siga las instrucciones de la interfaz que desea utilizar: • Para ello AWS CLI, consulte Autenticación con credencia les de usuario de IAM en la Guía del AWS Command Line Interface usuario. • Para obtener AWS SDKs información sobre las herramientas, consulte Autenticarse con credencia les de larga duración en la Guía de referencia de herramientas AWS SDKs y herramientas. • Para ello AWS APIs, consulte Administrar las claves de acceso para los usuarios de IAM en la Guía del usuario de IAM.

Uso de un perfil en su equipo local

Puedes ejecutar los ejemplos AWS CLI y códigos de esta guía con las credenciales a corto plazo que hayas creado. Ejecución del código en su equipo local Para obtener las credenciales y otra información de configuración, en los ejemplos se utiliza un perfil denominado `custom-labels-access`. Por ejemplo:

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

El usuario que representa el perfil debe tener permisos para llamar a las operaciones del SDK de Amazon Rekognition Custom Labels AWS y a otras operaciones del SDK que se necesitan en los

ejemplos. Para obtener más información, consulte Configuración de permisos de SDK. Para asignar permisos, consulte Configuración de permisos de SDK.

Para crear un perfil que funcione con los ejemplos de código AWS CLI y, elija una de las siguientes opciones. Asegúrese de que el nombre del perfil que haya creado es `custom-labels-access`.

- Usuarios administrados por IAM: siga las instrucciones que aparecen en Cambiar a un rol de IAM (AWS CLI).
- Identidad de la fuerza laboral (usuarios gestionados por AWS IAM Identity Center): siga las instrucciones que se indican en Configuración de la AWS CLI que va a utilizar AWS IAM Identity Center. Para los ejemplos de código, le recomendamos usar un entorno de desarrollo integrado (IDE), que sea compatible con el kit de herramientas de AWS y que permita la autenticación a través del IAM Identity Center. Para ver los ejemplos de Java, consulte Comenzar a crear con Java. Para ver los ejemplos de Java, consulte Comenzar a crear con Python. Para obtener más información, consulte Credenciales de IAM Identity Center.

> **Note**
>
> Puede usar el código para obtener las credenciales a corto plazo. Para obtener más información, consulte Cambiar a un rol de IAM (AWS API). En el caso del Identity Center IAM, consiga las credenciales a corto plazo de un rol siguiendo las instrucciones que se indican en Obtener las credenciales de rol de IAM para el acceso a la CLI.

Ejecutar código en AWS entornos

No debes usar las credenciales de usuario para firmar las llamadas al AWS SDK en AWS entornos, como el código de producción que se ejecuta en una AWS Lambda función. En su lugar, debe configurar un rol que defina los permisos que necesita el código. Tras esto, asocie la función al entorno en el que se ejecute el código. La forma de asignar el rol y hacer que las credenciales temporales estén disponibles varía en función del entorno en el que se ejecute el código:

- AWS Lambda función: utilice las credenciales temporales que Lambda proporciona automáticamente a la función cuando asume la función de ejecución de la función Lambda. Las credenciales están disponibles en las variables de entorno de Lambda. No es necesario especificar un perfil. Para obtener más información, consulte Rol de ejecución de Lambda.

- Amazon EC2 : usa el proveedor de credenciales de punto final de metadatos de EC2 instancias de Amazon. El proveedor genera y actualiza automáticamente las credenciales para ti mediante el perfil de EC2 instancia de Amazon que adjuntas a la EC2 instancia de Amazon. Para obtener más información, consulta Cómo usar un rol de IAM para conceder permisos a las aplicaciones que se ejecutan en instancias de Amazon EC2
- Amazon Elastic Container Service: utilice el proveedor de credenciales de Container. Amazon ECS envía y actualiza las credenciales a un punto de conexión de metadatos. Un rol de IAM de tarea que indique proporcionará una estrategia para administrar las credenciales que utilice su aplicación. Para obtener más información, consulte Interacción con servicios de AWS.

Para obtener más información sobre los proveedores de credenciales, consulte Proveedores de credenciales estandarizados.

Configuración de permisos de SDK

Para utilizar las operaciones del SDK de Etiquetas personalizadas de Amazon Rekognition, necesita permisos de acceso a la API de Etiquetas personalizadas de Amazon Rekognition y al bucket de Amazon S3 que se utilice para el entrenamiento de modelos.

Temas

- Otorgar permisos de operaciones del SDK
- Actualizaciones de políticas para el uso del AWS SDK
- Asignación de permisos

Otorgar permisos de operaciones del SDK

Se recomienda conceder solo los permisos necesarios para realizar una tarea (permisos de privilegios mínimos). Por ejemplo, para llamar DetectCustomLabels, necesitas permiso para actuar`rekognition:DetectCustomLabels`. Para buscar los permisos de una operación, consulte la referencia de API.

Cuando esté comenzando con una aplicación, es posible que no conozca los permisos concretos que necesite, por lo que puede empezar con los permisos más amplios. Las políticas administradas de AWS otorgan permisos que le servirán como punto de partida. Puede utilizar la política `AmazonRekognitionCustomLabelsFullAccess` AWS gestionada para obtener acceso completo a la API de etiquetas personalizadas de Amazon Rekognition. Para obtener más información,

consulte la política administrada de AWS: AmazonRekognitionCustomLabelsFullAccess. Cuando conozca los permisos que necesita su aplicación, cree políticas administradas por el cliente según la finalidad de uso para reducir aún más el número de permisos. Para obtener más información, consulte Políticas administradas por el cliente.

Para asignar permisos, consulte Asignación de permisos.

Actualizaciones de políticas para el uso del AWS SDK

Para usar el AWS SDK con la última versión de Amazon Rekognition Custom Labels, ya no necesita conceder permisos a Amazon Rekognition Custom Labels para acceder al bucket de Amazon S3 que contiene sus imágenes de formación y pruebas. Si ha añadido permisos anteriormente, no deberá eliminarlos. Si lo desea, elimine cualquier política del bucket donde el servicio de la entidad principal sea `rekognition.amazonaws.com`. Por ejemplo:

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

Para obtener más información, consulte Uso de políticas de buckets.

Asignación de permisos

Para dar acceso, agregue permisos a los usuarios, grupos o roles:

- Usuarios y grupos en: AWS IAM Identity Center

  Cree un conjunto de permisos. Siga las instrucciones de Creación de un conjunto de permisos en la Guía del usuario de AWS IAM Identity Center .
- Usuarios gestionados en IAM a través de un proveedor de identidades:

  Cree un rol para la federación de identidades. Siga las instrucciones descritas en Creación de un rol para un proveedor de identidad de terceros (federación) en la Guía del usuario de IAM.
- Usuarios de IAM:
  - Cree un rol que el usuario pueda aceptar. Siga las instrucciones descritas en Creación de un rol para un usuario de IAM en la Guía del usuario de IAM.
  - (No recomendado) Adjunte una política directamente a un usuario o añada un usuario a un grupo de usuarios. Siga las instrucciones descritas en Adición de permisos a un usuario (consola) de la Guía del usuario de IAM.

Llamar a una operación de Etiquetas personalizadas de Amazon Rekognition

Ejecute el siguiente código para confirmar que puede realizar llamadas a la API de Etiquetas personalizadas de Amazon Rekognition. El código muestra los proyectos de tu AWS cuenta, en la AWS región actual. Si no ha creado ningún proyecto anteriormente, la respuesta saldrá vacía, pero le confirmará que puede llamar a la operación `DescribeProjects`.

En general, para llamar a una función de ejemplo es necesario un cliente de Rekognition del AWS SDK y cualquier otro parámetro obligatorio. El cliente del AWS SDK se declara en la función principal.

Si el código no funciona, compruebe que el usuario tenga los permisos adecuados. Comprueba también que la AWS región que utilizas como etiquetas personalizadas de Amazon Rekognition no esté disponible en todas las regiones. AWS

Cómo llamar a una operación de Etiquetas personalizadas de Amazon Rekognition

1. Si aún no lo ha hecho, instale y configure el y el AWS CLI . AWS SDKs Para obtener más información, consulte Paso 4: Configure y AWS CLIAWS SDKs.
2. Utilice el siguiente código de ejemplo para ver sus proyectos.

   CLI

   Usa el comando `describe-projects` para ver los proyectos en su cuenta.

   ```
   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);
        }

    }
```

```
}
```

Paso 5: (Opcional) Cifrar los archivos de entrenamiento

Puede elegir una de las siguientes opciones para cifrar los archivos de manifiesto y los archivos de imagen de Etiquetas personalizadas de Amazon Rekognition que se encuentran en un bucket de consola o en un bucket externo de Amazon S3.

- Use una clave de Amazon S3 (SSE-S3).
- Usa tu AWS KMS key.

> **Note**
>
> El proceso de llamada a la entidad principal de IAM necesita una serie de permisos para descifrar los archivos. Para obtener más información, consulte Descifrar archivos cifrados con AWS Key Management Service.

Para obtener más información sobre el cifrado de un bucket de Amazon S3, consulte Ajuste del comportamiento predeterminado del cifrado del servidor para buckets de Amazon S3.

Descifrar archivos cifrados con AWS Key Management Service

Si utiliza AWS Key Management Service (KMS) para cifrar los archivos de manifiesto y los archivos de imagen de Amazon Rekognition Custom Labels, añada el principal de IAM que denomina Amazon Rekognition Custom Labels a la política de claves de la clave de KMS. De este modo, Etiquetas personalizadas de Amazon Rekognition podrá descifrar los archivos de manifiesto y de imagen antes del entrenamiento. Para obtener más información, consulte Mi bucket de Amazon S3 tiene cifrado predeterminado mediante una clave de AWS KMS personalizada. ¿Cómo puedo permitir que los usuarios descarguen y suban contenido al bucket?

La entidad principal de IAM necesita los siguientes permisos en la clave de KMS.

- kms: GenerateDataKey
- kms:Decrypt

Para obtener más información, consulte Protección de datos mediante el cifrado del servidor con claves de KMS almacenadas en AWS Key Management Service (SSE-KMS).

Cifrado de imágenes de entrenamiento y de prueba copiadas

Para entrenar el modelo, Etiquetas personalizadas de Amazon Rekognition hace una copia de las imágenes de entrenamiento y de prueba de origen. De forma predeterminada, las imágenes copiadas se cifran en reposo con una clave que AWS posee y administra. También puede optar por utilizar su propia AWS KMS key. Si usa su propia clave de KMS, necesitará los siguientes permisos en la clave de KMS.

- km: CreateGrant
- km: DescribeKey

Si lo desea, puede indicar la clave de KMS cuando entrene el modelo con la consola o cuando llama a la operación `CreateProjectVersion`. La clave de KMS que utilice no tiene por qué ser la misma clave de KMS usada para cifrar los archivos de manifiesto y de imagen en el bucket de Amazon S3. Para obtener más información, consulte Paso 5: (Opcional) Cifrar los archivos de entrenamiento.

Para obtener más información, consulte Conceptos de AWS Key Management Service. Las imágenes de origen no se ven afectadas.

Para obtener más información sobre el entrenamiento de un modelo, consulte Entrenamiento de un modelo de Etiquetas personalizadas de Amazon Rekognition.

Paso 6: (Opcional) Asociar conjuntos de datos anteriores a nuevos proyectos

Etiquetas personalizadas de Amazon Rekognition ahora administra conjuntos de datos con proyectos. Los conjuntos de datos previos (antiguos) que haya creado son de solo lectura y deben estar asociados a un proyecto antes de poder usarlos. Al abrir la página de detalles de un proyecto a través de la consola, asociamos automáticamente al proyecto los conjuntos de datos que entrenaron la última versión del modelo del proyecto. La asociación automática de un conjunto de datos con un proyecto no se produce si se utiliza el AWS SDK.

Los conjuntos de datos anteriores no asociados son los que no se han usado entrenar un modelo o se usaron para entrenar la versión anterior de un modelo. En la página de conjuntos de datos anteriores aparecen todos los conjuntos de datos asociados y no asociados.

Para usar un conjunto de datos anterior no asociado, cree un nuevo proyecto en la página de conjuntos de datos anteriores. El conjunto de datos se convertirá en el conjunto de datos de entrenamiento del nuevo proyecto. También puede crear un proyecto para un conjunto de datos ya asociado, ya que los conjuntos de datos anteriores pueden tener varias asociaciones.

Cómo asociar un conjunto de datos anterior a un proyecto nuevo

1. Abra la consola Amazon Rekognition en. https://console.aws.amazon.com/rekognition/
2. En el panel izquierdo, elija Usar etiquetas personalizadas. Se abrirá la página de inicio de Etiquetas personalizadas de Amazon Rekognition.
3. En el panel de navegación de la izquierda, elija Conjuntos de datos anteriores.
4. En la vista de conjuntos de datos, elija el conjunto de datos anterior que desee asociar a un proyecto.
5. Elija Crear proyecto con conjunto de datos.
6. En la ventana Crear un proyecto, en Nombre del proyecto, escriba un nombre para el proyecto.
7. Seleccione Crear proyecto para crearlo. La creación del proyecto puede tardar un tiempo.
8. Use el proyecto. Para obtener más información, consulte Qué es Etiquetas personalizadas de Amazon Rekognition.

Uso de un conjunto de datos anterior como conjunto de datos de prueba

Puede usar un conjunto de datos anterior como conjunto de datos de prueba en un proyecto existente asociando primero el conjunto de datos anterior a un proyecto nuevo. Tras esto, debe copiar el conjunto de datos de entrenamiento del nuevo proyecto al conjunto de datos de prueba del proyecto existente.

Cómo usar un conjunto de datos anterior como conjunto de datos de prueba

1. Siga las instrucciones que aparecen en Paso 6: (Opcional) Asociar conjuntos de datos anteriores a nuevos proyectos para asociar el conjunto de datos anterior a un proyecto nuevo.
2. Cree el conjunto de datos de prueba del proyecto existente copiando el conjunto de datos de entrenamiento del nuevo proyecto. Para obtener más información, consulte Copia de contenido de un conjunto de datos existente.
3. Siga las instrucciones que aparecen en Eliminación de un proyecto de Etiquetas personalizadas de Amazon Rekognition (consola) para eliminar el nuevo proyecto.

También tiene la opción de crear el conjunto de datos de prueba mediante el archivo de manifiesto del conjunto de datos anterior. Para obtener más información, consulte Creación de un archivo de manifiesto.

Qué es Etiquetas personalizadas de Amazon Rekognition

En esta sección, se ofrece una descripción general del flujo de trabajo para entrenar y usar un modelo de etiquetas personalizadas de Amazon Rekognition con la consola y el SDK. AWS

> **Note**
>
> Etiquetas personalizadas de Amazon Rekognition se encarga de administrar los conjuntos de datos de un proyecto. Puede crear conjuntos de datos para sus proyectos con la consola y el SDK. AWS Si ha utilizado anteriormente Etiquetas personalizadas de Amazon Rekognition, es posible que tenga que asociar sus conjuntos de datos antiguos a un proyecto nuevo. Para obtener más información, consulte Paso 6: (Opcional) Asociar conjuntos de datos anteriores a nuevos proyectos.

Temas

Cómo decidir el tipo de modelo

Primero debe decidir qué tipo de modelo quiere entrenar, lo que depende de sus objetivos empresariales. Por ejemplo, puede entrenar un modelo para que encuentre un logotipo en las publicaciones de las redes sociales, identifique sus productos en las estanterías de las tiendas o clasifique las piezas de una máquina en una línea de montaje.

Etiquetas personalizadas de Amazon Rekognition puede entrenar los siguientes tipos de modelos:

- Buscar objetos, escenas y conceptos
- Buscar ubicaciones de objetos
- Buscar ubicación de marcas

Para ayudarle a decidir qué tipo de modelo podría entrenar, Etiquetas personalizadas de Amazon Rekognition incluye algunos ejemplos de proyectos que puede utilizar. Para obtener más información, consulte Introducción a Etiquetas personalizadas de Amazon Rekognition.

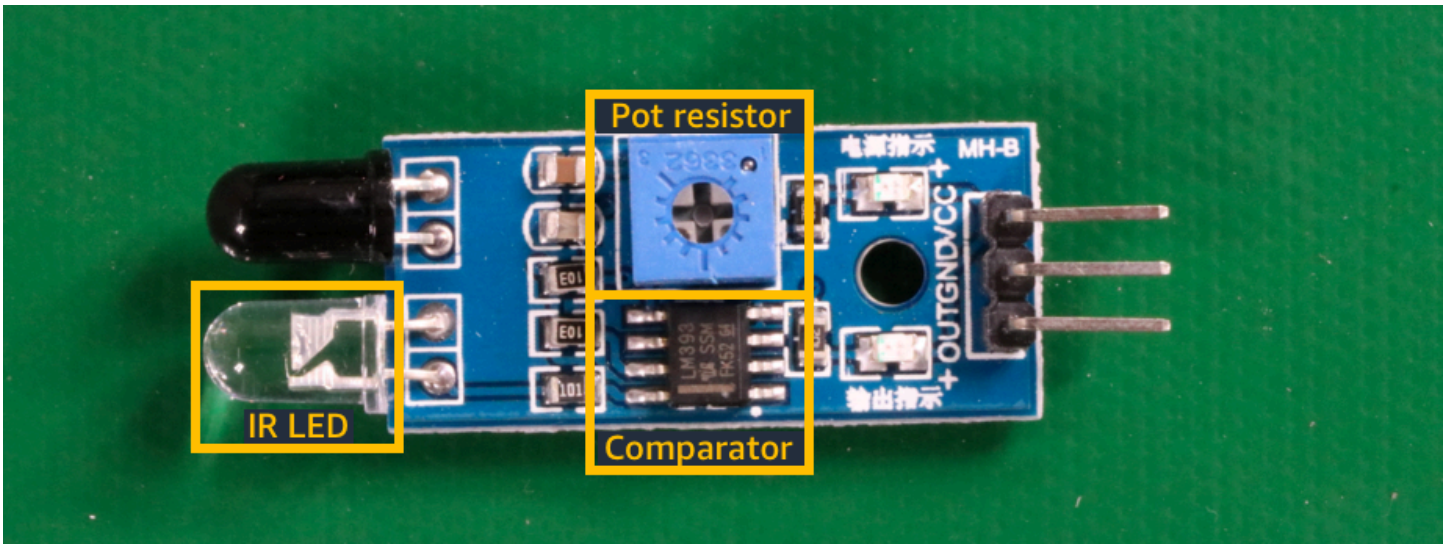
Buscar objetos, escenas y conceptos

El modelo clasifica los objetos, las escenas y los conceptos que están asociados a una imagen completa. Por ejemplo, puede entrenar un modelo para que determine si una imagen contiene una atracción turística o no. Para ver un proyecto de ejemplo, consulte Clasificación de imágenes. La siguiente imagen de un lago es un ejemplo del tipo de imagen en la que se pueden reconocer objetos, escenas y conceptos.

También tiene la opción de entrenar un modelo que clasifique las imágenes en varias categorías. Por ejemplo, la imagen anterior puede incluir categorías como el color del cielo, un reflejo o un lago. Para ver un proyecto de ejemplo, consulte Clasificación de imágenes de etiquetas múltiples.

Buscar ubicaciones de objetos

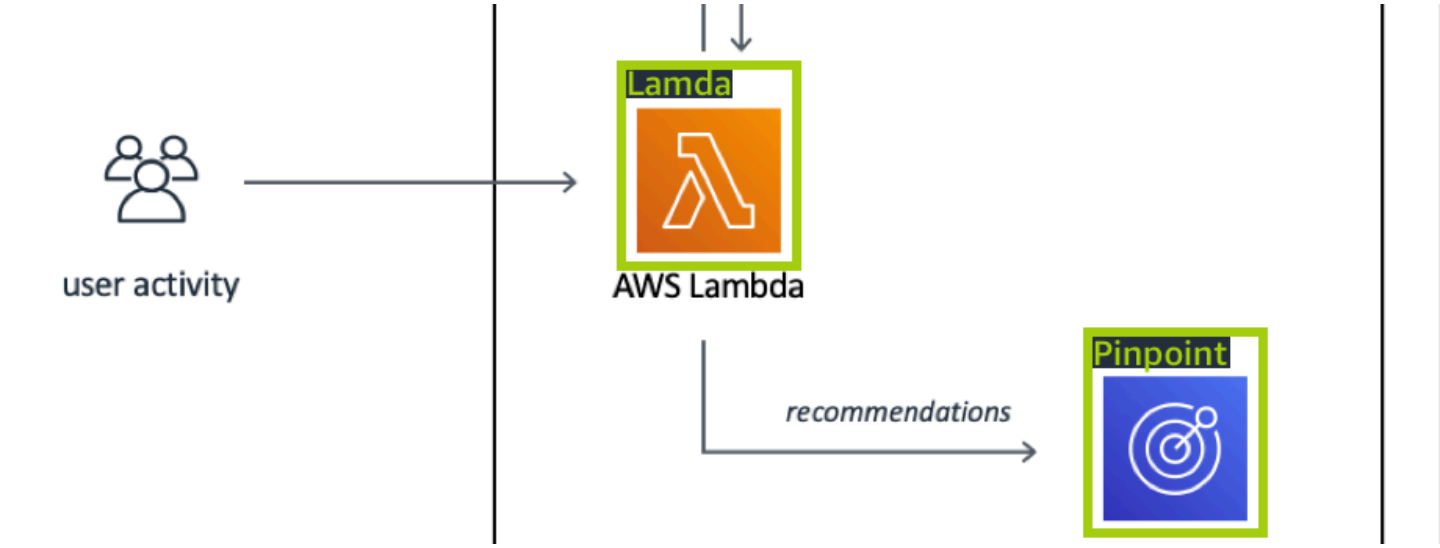
El modelo predice la ubicación de un objeto en una imagen. La predicción incluye información del cuadro delimitador en la ubicación del objeto y una etiqueta que identifica el objeto dentro del cuadro delimitador. Por ejemplo, en la siguiente imagen aparecen unos cuadros delimitadores alrededor de varios componentes de una placa de circuito, como un comparador o un potenciómetro.



En el proyecto de ejemplo Localización de objetos, se ve cómo Etiquetas personalizadas de Amazon Rekognition utiliza cuadros delimitadores etiquetados para entrenar un modelo que busca ubicaciones de objetos.

Buscar ubicación de marcas

Etiquetas personalizadas de Amazon Rekognition puede entrenar un modelo para que busque la ubicación de marcas, como los logotipos, en una imagen. La predicción incluye información del cuadro delimitador en la ubicación de la marca y una etiqueta que identifica el objeto dentro del cuadro delimitador. Para ver un proyecto de ejemplo, consulte Detección de marcas. La siguiente imagen es un ejemplo de algunas de las marcas que el modelo puede detectar.



Crear un modelo

El procedimiento para crear un modelo consiste en crear un proyecto, crear conjuntos de datos de entrenamiento y de prueba y entrenar el modelo.

Crear un proyecto

Un proyecto de Etiquetas personalizadas de Amazon Rekognition es un grupo de recursos necesarios para crear y administrar un modelo. El proyecto gestiona lo siguiente:

- Conjuntos de datos: las imágenes y las etiquetas de imagen que se utilizan para entrenar un modelo. Un proyecto incluye un conjunto de datos de entrenamiento y un conjunto de datos de prueba.
- Modelos: un modelo es un software que se entrena para buscar los conceptos, las escenas y los objetos que son exclusivos de su empresa o negocio. Puede haber varias versiones de un modelo en un proyecto.

Le recomendamos que utilice un proyecto para una sola aplicación práctica, como buscar componentes en una placa de circuito.

Puede crear un proyecto con la consola Amazon Rekognition Custom Labels y con la API. CreateProject Para obtener más información, consulte Creación de un proyecto.

Crear conjuntos de datos de entrenamiento y de prueba

Un conjunto de datos es un conjunto de imágenes y etiquetas que describen esas imágenes. Dentro del proyecto, se crea un conjunto de datos de entrenamiento y un conjunto de datos de prueba que Etiquetas personalizadas de Amazon Rekognition utiliza para entrenar y probar el modelo.

Una etiqueta identifica un objeto, una escena, un concepto o un cuadro delimitador alrededor del objeto de una imagen. Las etiquetas se asignan a una imagen completa (image-level) o se asignan a un cuadro delimitador que rodea el objeto de una imagen.

> **Important**
>
> La forma en que etiquete las imágenes en sus conjuntos de datos determina el tipo de modelo que se crea en Etiquetas personalizadas de Amazon Rekognition. Por ejemplo, para entrenar un modelo que busca objetos, escenas y conceptos, debe asignar etiquetas de imagen a las imágenes de sus conjuntos de datos de entrenamiento y de prueba. Para obtener más información, consulte Finalidad de los conjuntos de datos.

Las imágenes deben estar en formato PNG y JPEG y se deben seguir las recomendaciones de las imágenes de entrada. Para obtener más información, consulte Preparación de imágenes.

Crear conjuntos de datos de entrenamiento y de prueba (consola)

Puede empezar con un proyecto que tenga un único conjunto de datos o un proyecto que tenga conjuntos de datos de entrenamiento y de prueba independientes. Si empieza con un único conjunto de datos, Etiquetas personalizadas de Amazon Rekognition lo divide durante el entrenamiento para crear un conjunto de datos de entrenamiento (80 %) y un conjunto de datos de prueba (20 %) para su proyecto. Comience con un único conjunto de datos si quiere que Etiquetas personalizadas de Amazon Rekognition decida qué imágenes se van a usar para el entrenamiento y las pruebas. Para tener un control total sobre el entrenamiento, las pruebas y el nivel de rendimiento, le recomendamos que inicie el proyecto con conjuntos de datos de entrenamiento y de prueba independientes.

Para crear los conjuntos de datos para un proyecto, importe las imágenes a través de uno de los siguientes métodos:

- Importar imágenes de un ordenador local.
- Importar imágenes de un bucket de S3. Etiquetas personalizadas de Amazon Rekognition puede etiquetar las imágenes con los nombres de las carpetas que contienen las imágenes.

- Importa un archivo de manifiesto de Amazon SageMaker AI Ground Truth.
- Copiar un conjunto de datos existente de Etiquetas personalizadas de Amazon Rekognition.

Para obtener más información, consulte Creación de conjuntos de datos de entrenamiento y prueba.

Según el lugar desde el que importe las imágenes, es posible que no vengan etiquetadas. Por ejemplo, las imágenes importadas de un ordenador local no están etiquetadas. Las imágenes importadas de un archivo de manifiesto de Amazon SageMaker AI Ground Truth están etiquetadas. Puede utilizar la consola de Etiquetas personalizadas de Amazon Rekognition para agregar, cambiar y asignar etiquetas. Para obtener más información, consulte Etiquetado de imágenes.

Para crear sus conjuntos de datos de entrenamiento y de prueba en la consola, consulte Creación de conjuntos de datos de entrenamiento y prueba. Si quiere ver el tutorial sobre cómo crear conjuntos de datos de entrenamiento y de prueba, consulte Clasificación de imágenes.

Crear conjuntos de datos de entrenamiento y prueba (SDK)

Puede crear sus conjuntos de datos de entrenamiento y de prueba, use la API de `CreateDataset`. Puede crear un conjunto de datos mediante un archivo de manifiesto en formato Amazon SageMaker o copiando un conjunto de datos de Etiquetas personalizadas de Amazon Rekognition existente. Para obtener más información, consulte Crear conjuntos de datos de entrenamiento y prueba (SDK) . Si es necesario, puede crear su propio archivo de manifiesto. Para obtener más información, consulte the section called “Creación de un archivo de manifiesto”.

Entrenar su modelo

Entrene su modelo con el conjunto de datos de entrenamiento. Se creará una nueva versión del modelo cada vez que se entrena. Durante el entrenamiento, Etiquetas personalizadas de Amazon Rekognition comprueba el rendimiento del modelo entrenado. Puede utilizar los resultados para evaluar y mejorar el modelo. El entrenamiento tarda un tiempo en realizarse. Solo se le cobrará cuando termine de entrenarse por completo un modelo. Para obtener más información, consulte Entrenamiento de un modelo de Etiquetas personalizadas de Amazon Rekognition. Si el entrenamiento del modelo presenta problemas, Etiquetas personalizadas de Amazon Rekognition le dará información sobre la depuración de errores que puede utilizar. Para obtener más información, consulte Depuración de un modelo de entrenamiento con errores.

Entrenar el modelo (consola)

Para entrenar su modelo a través de la consola, consulte Entrenamiento de un modelo (consola).

Entrenamiento de un modelo (SDK)

Puedes entrenar a un modelo de Amazon Rekognition Custom Labels llamando. CreateProjectVersion Para obtener más información, consulte Entrenamiento de un modelo (SDK).

Mejorar el modelo

Durante las pruebas, Etiquetas personalizadas de Amazon Rekognition genera métricas de evaluación que puede usar para mejorar el modelo entrenado.

Evaluar el modelo

Evalúe el rendimiento del modelo mediante las métricas de rendimiento creadas durante las pruebas. Las métricas de rendimiento, como la puntuación F1, las de precisión y las de exhaustividad, le permiten conocer el rendimiento del modelo entrenado y decidir si es posible usarlo en la fase de producción. Para obtener más información, consulte Métricas para evaluar su modelo.

Evaluar un modelo (consola)

Para ver las métricas de rendimiento, consulte Acceso a las métricas de evaluación (consola).

Evaluar un modelo (SDK)

Para obtener las métricas de rendimiento, llama DescribeProjectVersionspara obtener los resultados de las pruebas. Para obtener más información, consulte Acceso a las métricas de evaluación de Etiquetas personalizadas de Amazon Rekognition (SDK) . Los resultados de las pruebas se representan con métricas que no están disponibles en la consola, como una matriz de confusión para los resultados de la clasificación. Los resultados de las pruebas se devuelven en los siguientes formatos:

- Puntuación F1: un valor único que refleja el rendimiento general de precisión y exhaustividad del modelo. Para obtener más información, consulte F1.
- Ubicación del archivo de resumen: el resumen de las pruebas incluye una serie de métricas de evaluación acumuladas de todo el conjunto de datos de pruebas y las métricas de cada etiqueta por separado. `DescribeProjectVersions` devuelve la ubicación del bucket de S3 y de la carpeta del archivo de resumen. Para obtener más información, consulte Acceder al archivo de resumen del modelo.
- Ubicación del resumen del manifiesto de evaluación: este resumen incluye los detalles sobre los resultados de las pruebas, como los índices de confianza y los resultados de las pruebas

de clasificación binaria, como los falsos positivos. `DescribeProjectVersions` devuelve la ubicación del bucket de S3 y de la carpeta de los archivos de resumen. Para obtener más información, consulte Interpretación de la instantánea del manifiesto de evaluación.

Mejorar el modelo

Si se necesitan mejoras, puede agregar más imágenes de entrenamiento o mejorar el etiquetado de los conjuntos de datos. Para obtener más información, consulte Mejora de un modelo de Etiquetas personalizadas de Amazon Rekognition. También puede aportar sus observaciones sobre las predicciones que haga su modelo y utilizarlos para mejorarlo. Para obtener más información, consulte Mejora de un modelo con Model feedback.

Mejorar el modelo (consola)

Para agregar imágenes a un conjunto de datos, consulte Agregar más imágenes a un conjunto de datos. Para agregar o cambiar etiquetas, consulte the section called “Etiquetado de imágenes”.

Para volver a entrenar el modelo, consulte Entrenamiento de un modelo (consola).

Mejorar el modelo (SDK)

Para agregar imágenes a un conjunto de datos o cambiar el etiquetado de una imagen, usa la API de `UpdateDatasetEntries`. `UpdateDatasetEntries` actualiza o agrega líneas JSON a un archivo de manifiesto. Cada línea JSON contiene la información de una sola imagen, como las etiquetas asignadas o la información de los cuadros delimitadores. Para obtener más información, consulte Agregar más imágenes (SDK). Para ver las entradas de un conjunto de datos, usa la API de `ListDatasetEntries`.

Para volver a entrenar el modelo, consulte Entrenamiento de un modelo (SDK).

Iniciar el modelo

Antes de poder usar el modelo, debe iniciarlo a través de la consola de Etiquetas personalizadas de Amazon Rekognition o la API de `StartProjectVersion`. Se le cobrará por la cantidad de tiempo en que se ejecute el modelo. Para obtener más información, consulte Ejecución de un modelo entrenado.

Iniciar el modelo (consola)

Para iniciar el modelo con la consola, consulte Inicio de un modelo de Etiquetas personalizadas de Amazon Rekognition (consola).

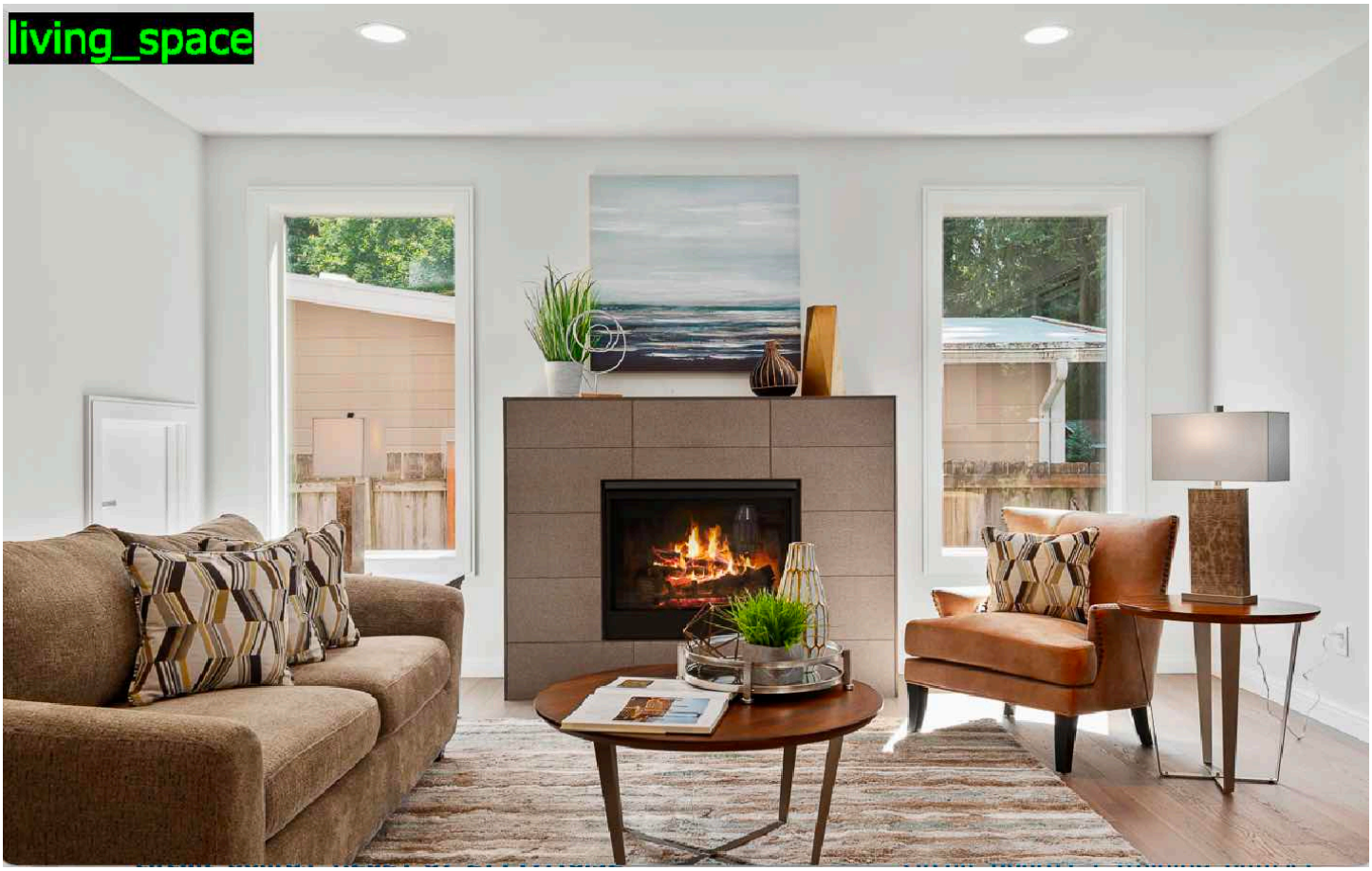
Iniciar el modelo

Empiezas a llamar a tu modelo StartProjectVersion. Para obtener más información, consulte Inicio de un modelo de Etiquetas personalizadas de Amazon Rekognition (SDK).

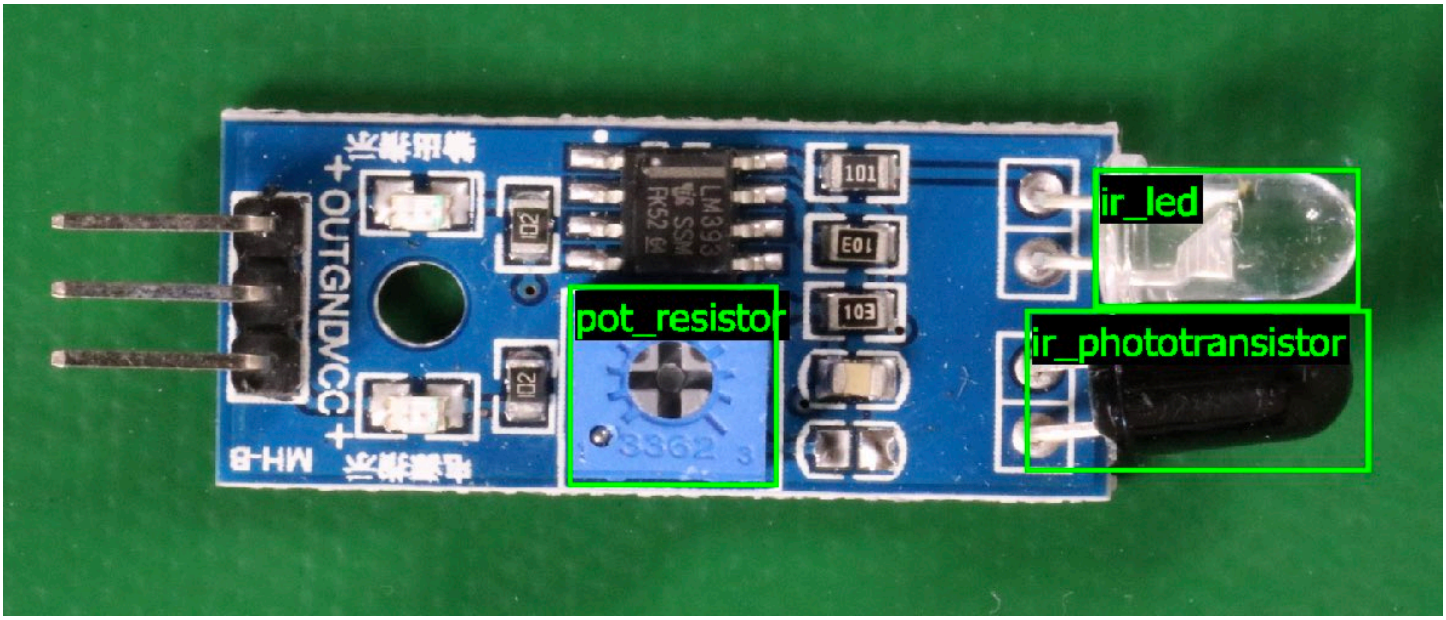
Analizar una imagen

Para analizar una imagen con el modelo, use la API de `DetectCustomLabels`. Puede indicar una imagen local o una imagen almacenada en un bucket de S3. La operación también requiere el nombre de recurso de Amazon (ARN) del modelo que desee utilizar.

Si el modelo encuentra objetos, escenas y conceptos, la respuesta incluirá una lista de etiquetas de imagen que se encuentran en la imagen. Por ejemplo, la siguiente imagen muestra las etiquetas de imagen que se encuentran en el proyecto de ejemplo Habitaciones.



Si el modelo encuentra ubicaciones de objetos, la respuesta incluirá una lista de los cuadros delimitadores etiquetados que se encuentran en la imagen. Un cuadro delimitador representa la ubicación de un objeto en una imagen. Puede utilizar la información del cuadro delimitador para dibujar un cuadro delimitador alrededor de un objeto. Por ejemplo, en la siguiente imagen aparecen unos cuadros delimitadores alrededor de los componentes de una placa de circuito que se han encontrado por medio del proyecto de ejemplo Placas de circuito.



Para obtener más información, consulte Análisis de una imagen con un modelo entrenado.

Detener el modelo

Se le cobrará por el tiempo de ejecución del modelo. Si ya no va a usar el modelo, deténgalo en la consola de Etiquetas personalizadas de Amazon Rekognition o mediante la API de `StopProjectVersion`. Para obtener más información, consulte Detención de un modelo de Etiquetas personalizadas de Amazon Rekognition.

Detener el modelo (consola)

Para detener la ejecución de un modelo con la consola, consulte Detención de un modelo de Etiquetas personalizadas de Amazon Rekognition (consola).

Detener el modelo (SDK)

Para detener a una modelo en marcha, llama StopProjectVersion. Para obtener más información, consulte Detención de un modelo de Etiquetas personalizadas de Amazon Rekognition (SDK).

Introducción a Etiquetas personalizadas de Amazon Rekognition

Antes de empezar con la Introducción, le recomendamos que lea Qué es Etiquetas personalizadas de Amazon Rekognition.

Etiquetas personalizadas de Amazon Rekognition sirve para entrenar un modelo de machine learning. El modelo entrenado analiza las imágenes para encontrar objetos, escenas y objetos que son exclusivos de su empresa o negocio. Por ejemplo, puede entrenar un modelo para que clasifique imágenes de casas o busque la ubicación de los componentes electrónicos en una placa de circuito impreso.

Para ayudarle a empezar, hay disponibles tutoriales de vídeo y proyectos de ejemplo de Etiquetas personalizadas de Amazon Rekognition.

> **Note**
>
> Para obtener información sobre AWS las regiones y los puntos de enlace compatibles con las etiquetas personalizadas de Amazon Rekognition, consulte Puntos de enlace y cuotas de Rekognition.

Tutoriales de vídeo

En los vídeos se explica cómo utilizar Etiquetas personalizadas de Amazon Rekognition para entrenar y utilizar un modelo.

Cómo ver los tutoriales de vídeo

1. Inicie sesión en la consola Amazon Rekognition AWS Management Console y ábrala en. https://console.aws.amazon.com/rekognition/
2. En el panel izquierdo, elija Usar etiquetas personalizadas. Se abrirá la página de inicio de Etiquetas personalizadas de Amazon Rekognition. Si no aparece la opción Utilizar etiquetas personalizadas, compruebe que la región de AWS que utiliza es compatible con Etiquetas personalizadas de Amazon Rekognition.
3. En el panel de navegación, elija Introducción.

4. En ¿Qué es Etiquetas personalizadas de Amazon Rekognition?, seleccione el vídeo para ver la introducción.
5. En el panel de navegación, elija Tutoriales.
6. En la página Tutoriales, elija los tutoriales de vídeo que desee ver.

Proyectos de ejemplo

Etiquetas personalizadas de Amazon Rekognition incluye los siguientes proyectos de ejemplo.

Clasificación de imágenes

El proyecto de clasificación de imágenes (habitaciones) entrena un modelo que busca una o varias ubicaciones de una casa en una imagen, como la terraza, la cocina y el patio. Las imágenes de entrenamiento y de prueba representan una única ubicación. Cada imagen está etiquetada con una única etiqueta de imagen, como cocina, patio o sala_estar. En el caso de una imagen analizada, el modelo entrenado devuelve una o varias etiquetas similares del conjunto de etiquetas de imagen utilizadas para el entrenamiento. Por ejemplo, el modelo podría encontrar la etiqueta sala_estar en la siguiente imagen. Para obtener más información, consulte Encontrar objetos, escenas y conceptos.

Clasificación de imágenes de etiquetas múltiples

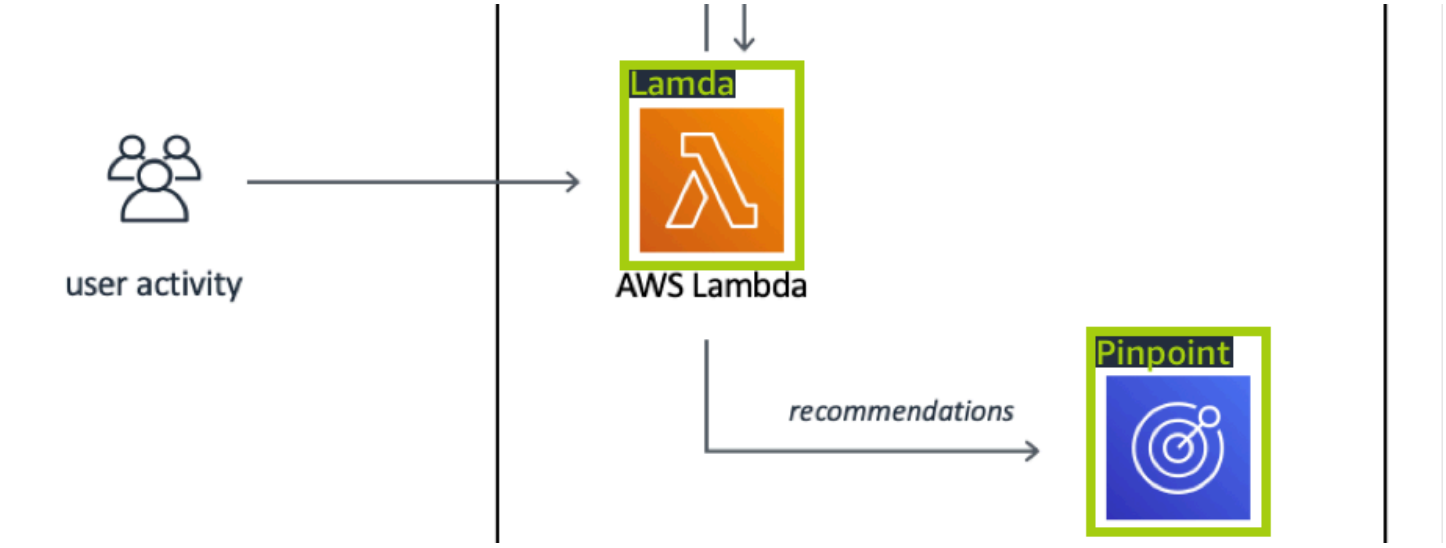
El proyecto de clasificación de imágenes con etiquetas múltiples (flores) entrena un modelo que clasifica las imágenes de flores en tres conceptos (tipo de flor, presencia de hojas y fase de crecimiento).

Las imágenes de entrenamiento y de prueba tienen etiquetas de imagen por cada concepto, como camelia para un tipo de flor, con_hojas para una flor con hojas y floración_total para una flor que ya ha crecido.

En el caso de una imagen analizada, el modelo entrenado devuelve etiquetas similares del conjunto de etiquetas de imagen utilizadas para el entrenamiento. Por ejemplo, el modelo devuelve las etiquetas euforbio_mediterráneo y con_hojas para la siguiente imagen. Para obtener más información, consulte Encontrar objetos, escenas y conceptos.

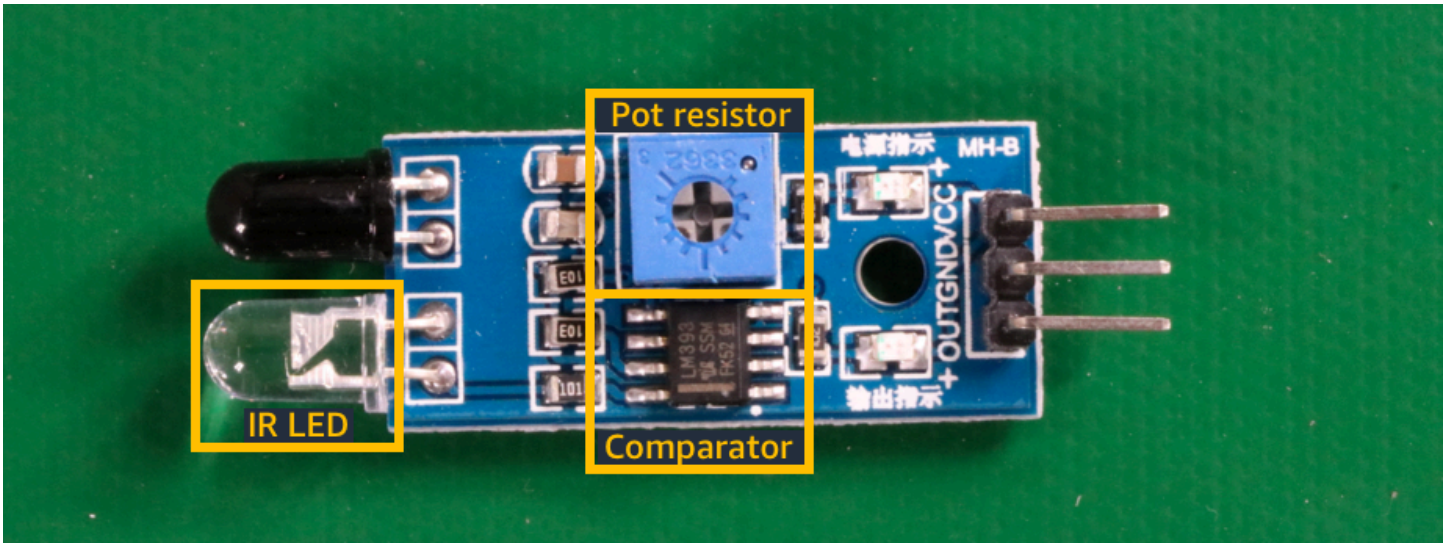
Detección de marcas

El proyecto de detección de marcas (Logos) entrena a un modelo que encuentra la ubicación de determinados AWS logotipos, como Amazon Textract y AWS lambda. Las imágenes de entrenamiento son solo de logotipo y tienen una única etiqueta de imagen, como lambda o textract. También es posible entrenar un modelo de detección de marcas con imágenes de entrenamiento que tengan cuadros delimitadores para las ubicaciones de las marcas. Las imágenes de prueba tienen cuadros delimitadores etiquetados que representan la ubicación de los logotipos en ubicaciones naturales, como un diagrama arquitectónico. El modelo entrenado busca los logotipos y devuelve un cuadro delimitador etiquetado por cada logotipo encontrado. Para obtener más información, consulte Encontrar ubicaciones de marcas.



Localización de objetos

El proyecto de localización de objetos (placas de circuito) entrena un modelo que busca la ubicación de las piezas en una placa de circuito impreso, como un comparador o un diodo emisor de luz infrarroja. Las imágenes de entrenamiento y prueba incluyen cuadros delimitadores que rodean los componentes de la placa de circuito y una etiqueta que identifica la pieza dentro del cuadro delimitador. En la siguiente imagen de ejemplo, los nombres de las etiquetas son fototransistor_ir, led_ir, pot_resistor y comparator. El modelo entrenado busca los componentes de la placa de circuito y devuelve un cuadro delimitador etiquetado por cada pieza del circuito encontrada. Para obtener más información, consulte Encontrar ubicaciones de objetos.



Aplicación de los proyectos de ejemplo

Estas instrucciones de primeros pasos le explican cómo entrenar un modelo mediante proyectos de ejemplo que Etiquetas personalizadas de Amazon Rekognition crea para usted. También le indica cómo iniciar el modelo y usarlo para analizar una imagen.

Creación del proyecto de ejemplo

Para empezar, decida qué proyecto usar. Para obtener más información, consulte Paso 1: Elegir un proyecto de ejemplo.

Etiquetas personalizadas de Amazon Rekognition utiliza conjuntos de datos para entrenar y evaluar (probar) un modelo. Un conjunto de datos administra las imágenes y las etiquetas que identifican el contenido de las imágenes. Los proyectos de ejemplo incluyen un conjunto de datos de

entrenamiento y un conjunto de datos de prueba en los que se etiquetan todas las imágenes. No es necesario hacer ningún cambio antes de entrenar el modelo. En los proyectos de ejemplo se dan las dos formas en que Etiquetas personalizadas de Amazon Rekognition utiliza etiquetas para entrenar distintos tipos de modelos.

- image-level: la etiqueta identifica un objeto, escena o concepto que representa la imagen completa.
- bounding box: la etiqueta identifica el contenido de un cuadro delimitador. Un cuadro delimitador es un conjunto de coordenadas de imagen que rodean el objeto de una imagen.

Más adelante, cuando cree un proyecto con sus propias imágenes, deberá crear conjuntos de datos de entrenamiento y prueba y también etiquetar las imágenes. Para obtener más información, consulte Cómo decidir el tipo de modelo.

Entrenamiento de un modelo

Una vez que Etiquetas personalizadas de Amazon Rekognition haya creado el proyecto de ejemplo, puede entrenar el modelo. Para obtener más información, consulte Paso 2: Entrenar un modelo. Una vez finalizado el entrenamiento, normalmente se evalúa el rendimiento del modelo. Las imágenes del conjunto de datos de ejemplo ya crean un modelo de alto rendimiento y no es necesario evaluarlo antes de ejecutarlo. Para obtener más información, consulte Mejora de un modelo de Etiquetas personalizadas de Amazon Rekognition.

Uso del modelo

A continuación, inicie el modelo. Para obtener más información, consulte Paso 3: Ejecutar el modelo.

Después de empezar a ejecutar el modelo, puede usarlo para analizar nuevas imágenes. Para obtener más información, consulte Paso 4: Analizar una imagen con su modelo.

Se le cobrará por la cantidad de tiempo de ejecución del modelo. Cuando termine de usar el modelo de ejemplo, deberá detenerlo. Para obtener más información, consulte Paso 5: Detener el modelo.

Pasos a seguir a continuación

Cuando lo tenga todo listo, puede crear e implementar sus propios proyectos. Para obtener más información, consulte Paso 6: Siguientes pasos.

Paso 1: Elegir un proyecto de ejemplo

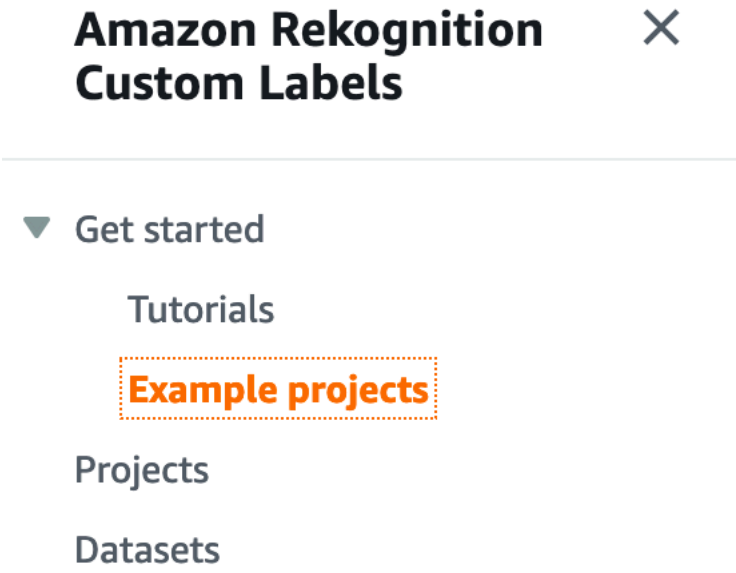
En este paso, elija un proyecto de ejemplo. A continuación, Etiquetas personalizadas de Amazon Rekognition crea un proyecto y un conjunto de datos para usted. El proyecto sirve para administrar los archivos que se utilizan para entrenar el modelo. Para obtener más información, consulte Administración de un proyecto de Etiquetas personalizadas de Amazon Rekognition. Los conjuntos de datos contienen las imágenes, las etiquetas asignadas y los cuadros delimitadores que se utilizan para entrenar y probar un modelo. Para obtener más información, consulte the section called "Administración de conjuntos de datos".

Para obtener más información acerca de los proyectos de ejemplo, consulte Proyectos de ejemplo.

Elegir un proyecto de ejemplo

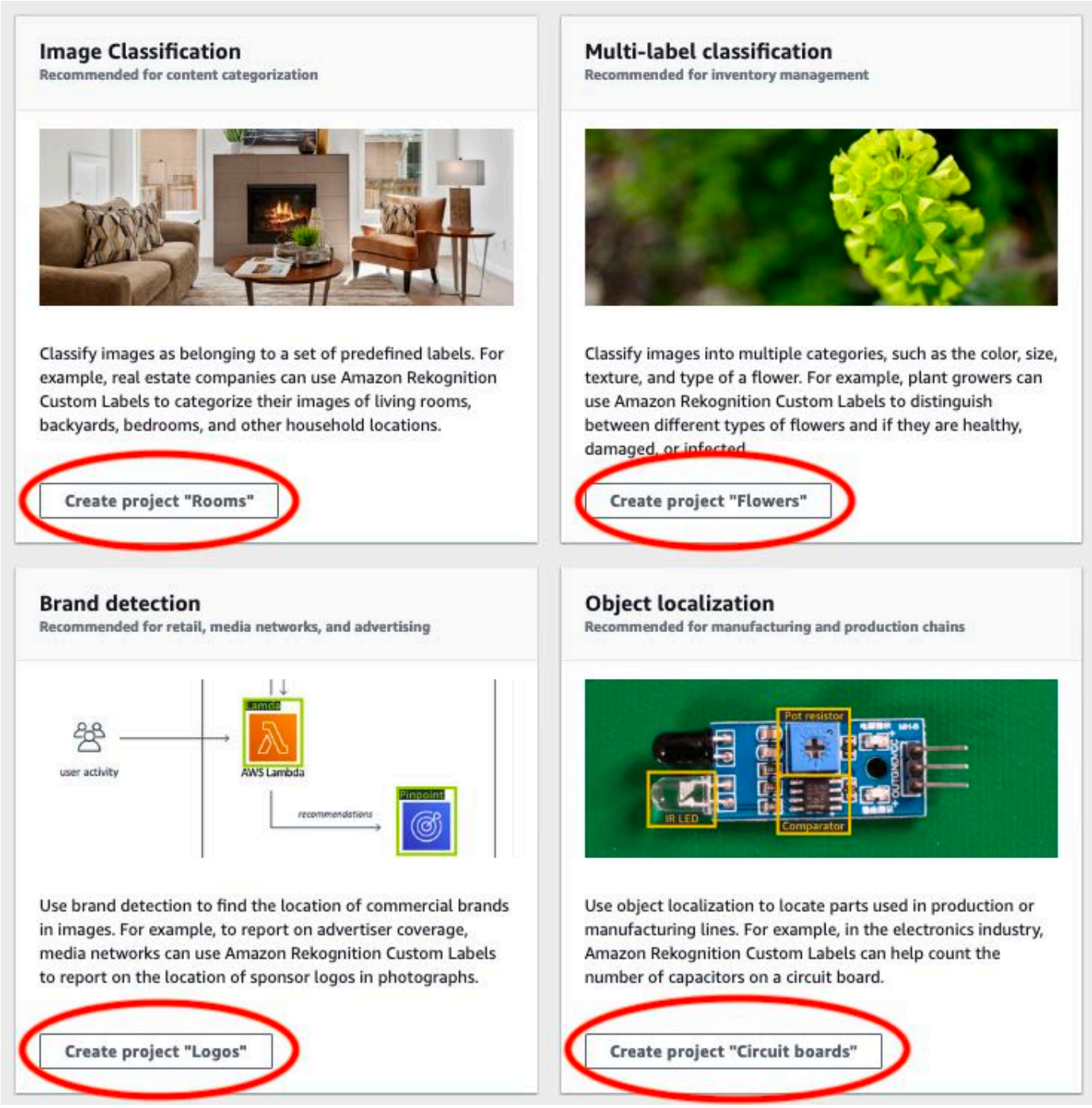
1. Inicie sesión en la consola Amazon Rekognition AWS Management Console y ábrala en. https://console.aws.amazon.com/rekognition/
2. En el panel izquierdo, elija Usar etiquetas personalizadas. Se abrirá la página de inicio de Etiquetas personalizadas de Amazon Rekognition. Si no aparece la opción Utilizar etiquetas personalizadas, compruebe que la región de AWS que utiliza es compatible con Etiquetas personalizadas de Amazon Rekognition.
3. Elija Comenzar.

   Sección Etiquetas personalizadas de Amazon Rekognition donde se muestra cómo empezar, tutoriales con "proyectos de ejemplo" destacados, proyectos y conjuntos de datos.

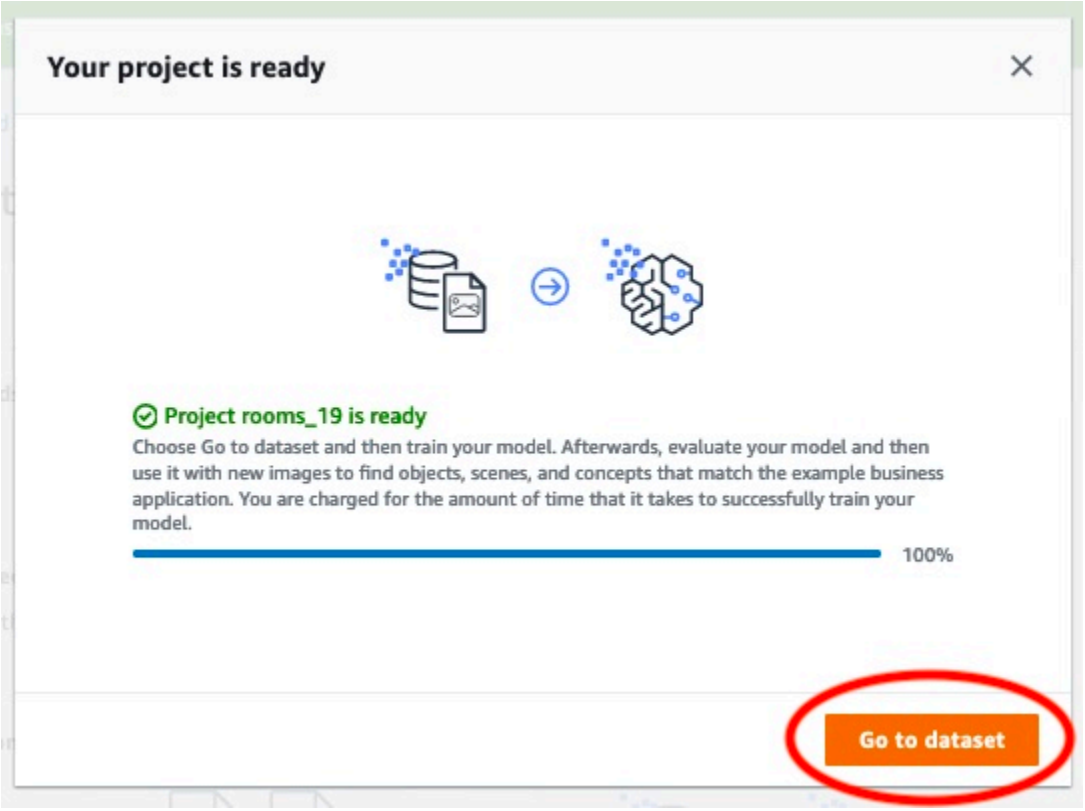


4. En Examinar proyectos de ejemplo, elija Probar proyectos de ejemplo.
5. Decida qué proyecto quiere usar y elija Crear proyecto ***`project name`*** "» en la sección de ejemplos. Tras esto, Etiquetas personalizadas de Amazon Rekognition creará un proyecto de ejemplo para usted.

> **Note**
>
> Si es la primera vez que abre la consola en la AWS región actual, aparecerá el cuadro de diálogo de configuración por primera vez. Haga lo siguiente:
>
> 1. Anote el nombre del bucket de Amazon S3 que aparece.
> 2. Seleccione Continuar para permitir que Etiquetas personalizadas de Amazon Rekognition cree un bucket de Amazon S3 (bucket de consola) en su nombre. La siguiente imagen de la consola muestra ejemplos con los botones “Crear proyecto” para la clasificación de imágenes (habitaciones), la clasificación con varias etiquetas (flores), la detección de marcas (logotipos) y la localización de objetos (placas de circuitos).



6. Cuando el proyecto esté listo, elija Ir a conjunto de datos. En la imagen siguiente, se muestra el aspecto del panel del proyecto cuando el proyecto está listo.

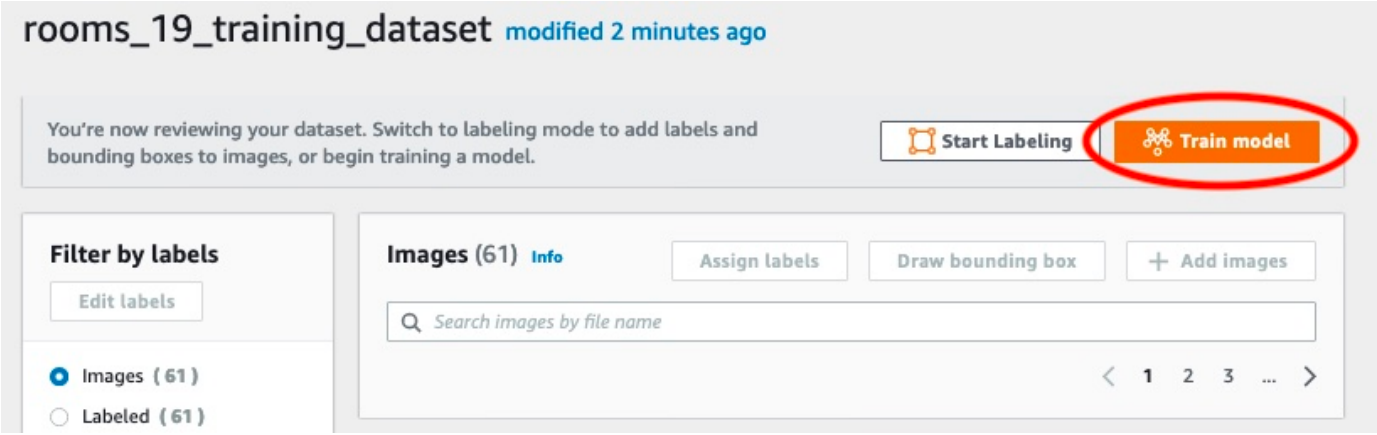


Paso 2: Entrenar un modelo

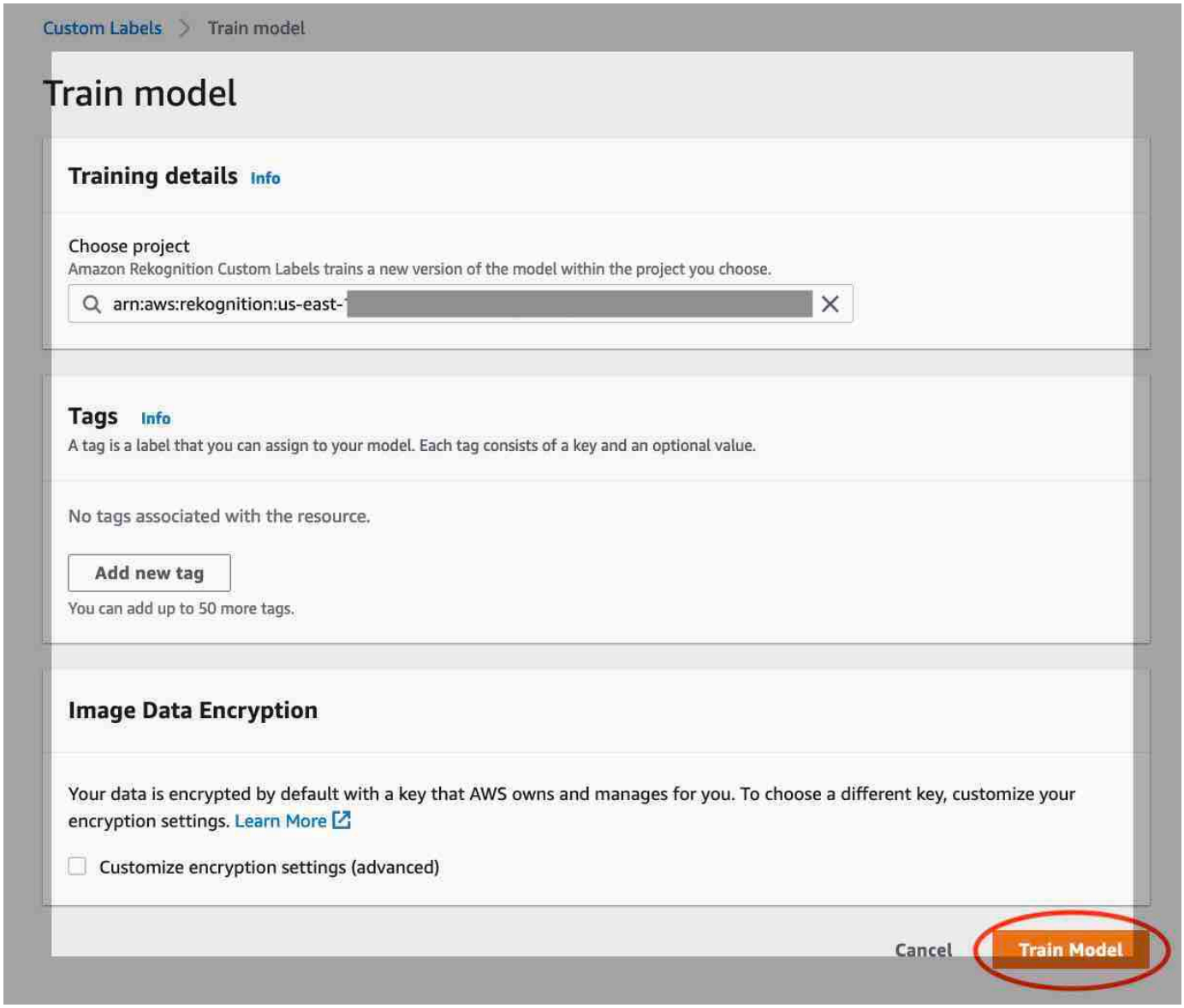
En este paso entrenará su modelo. Los conjuntos de datos de entrenamiento y de prueba se configuran automáticamente por usted. Una vez que el entrenamiento se complete correctamente, podrá ver los resultados generales de la evaluación y los resultados de las evaluaciones de cada una de las imágenes de prueba. Para obtener más información, consulte Entrenamiento de un modelo de Etiquetas personalizadas de Amazon Rekognition.

Cómo entrenar su modelo

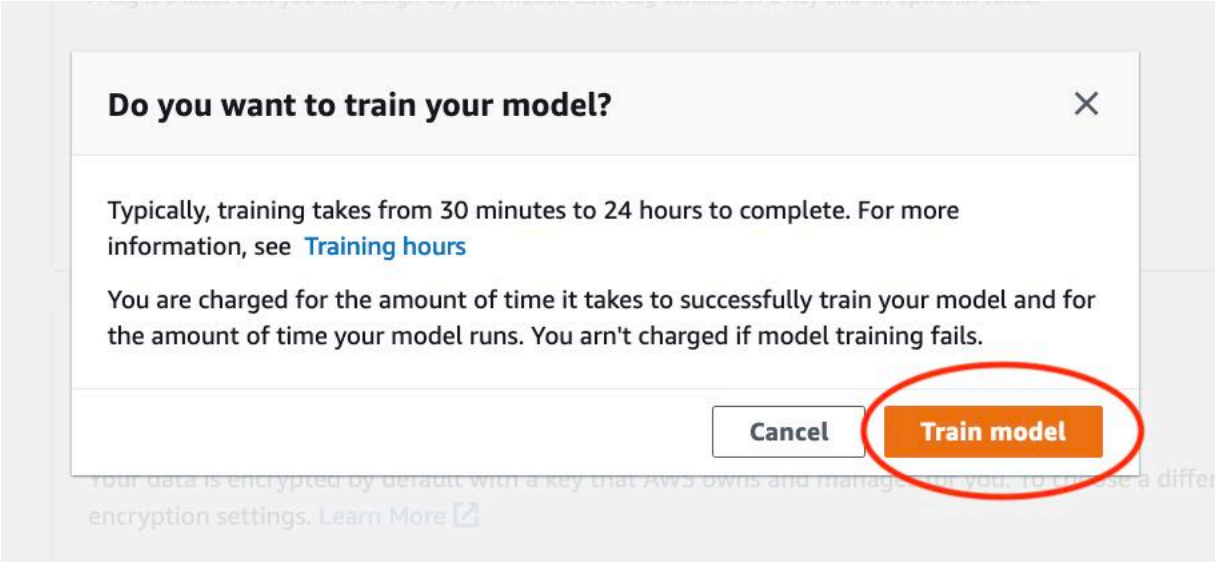
1. En la página del conjunto de datos, seleccione Entrenar modelo. En la imagen siguiente, se muestra la consola con el botón Entrenar modelo.



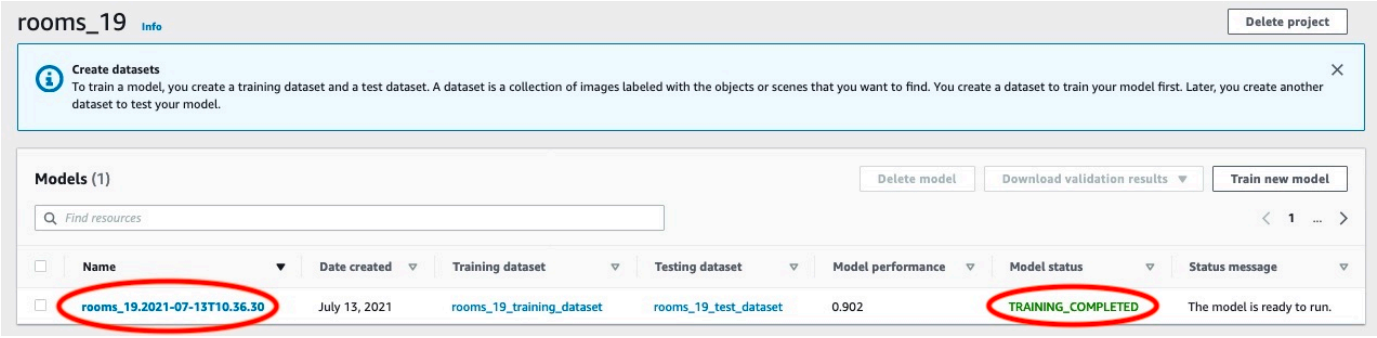
2. En la página Entrenar modelo, elija Entrenar modelo. En la imagen siguiente se muestra el botón Entrenar modelo; observe que el nombre de recurso de Amazon (ARN) del proyecto se encuentra en el cuadro de edición Elegir un proyecto.



3. En el cuadro de diálogo ¿Quiere entrenar su modelo?, que se muestra en la página siguiente, elija Entrenar modelo.

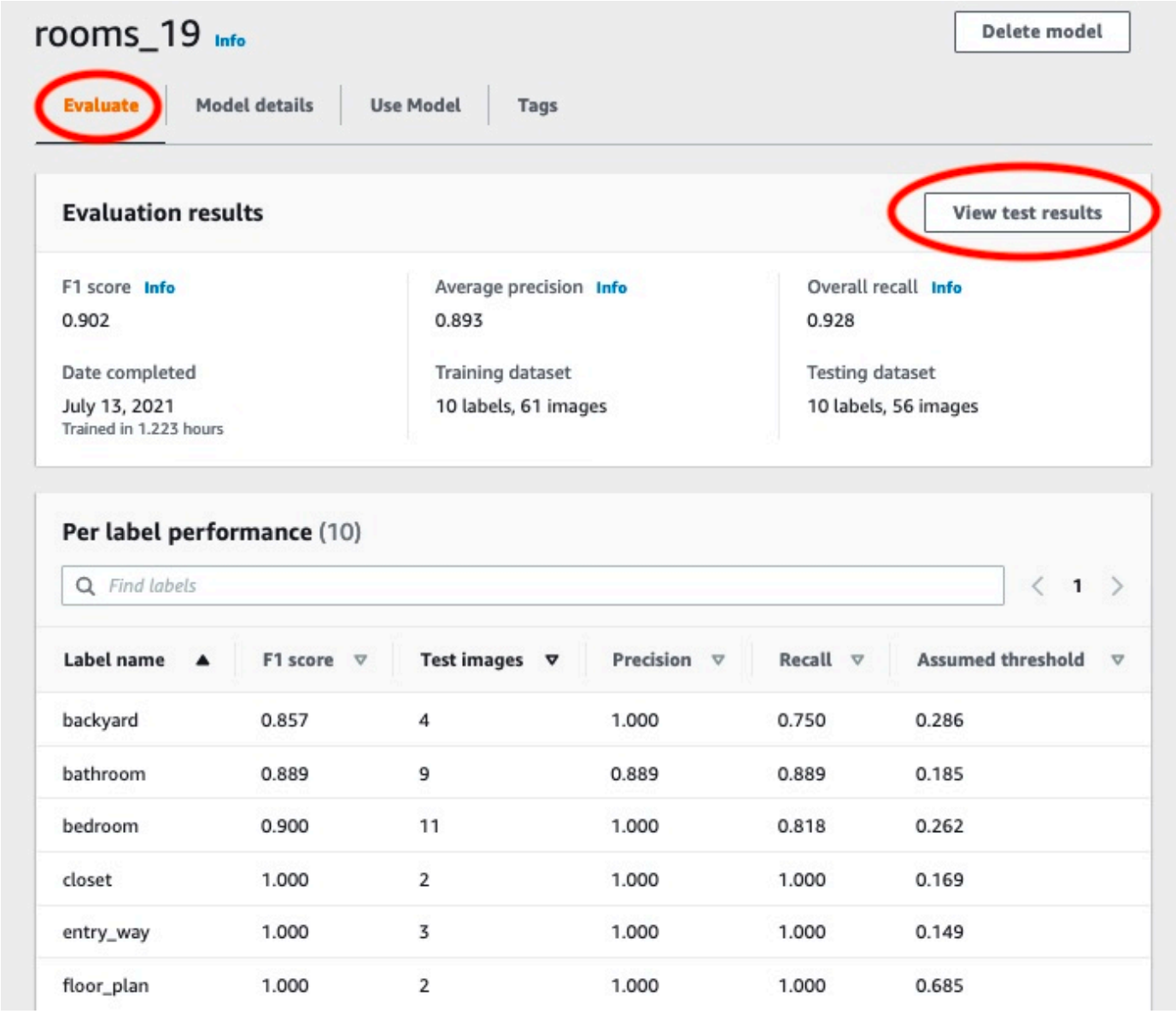


4. Una vez finalizado, elija el nombre del modelo. El entrenamiento finaliza cuando el estado del modelo es TRAINING_COMPLETED, como se muestra en la siguiente captura de pantalla de la consola.

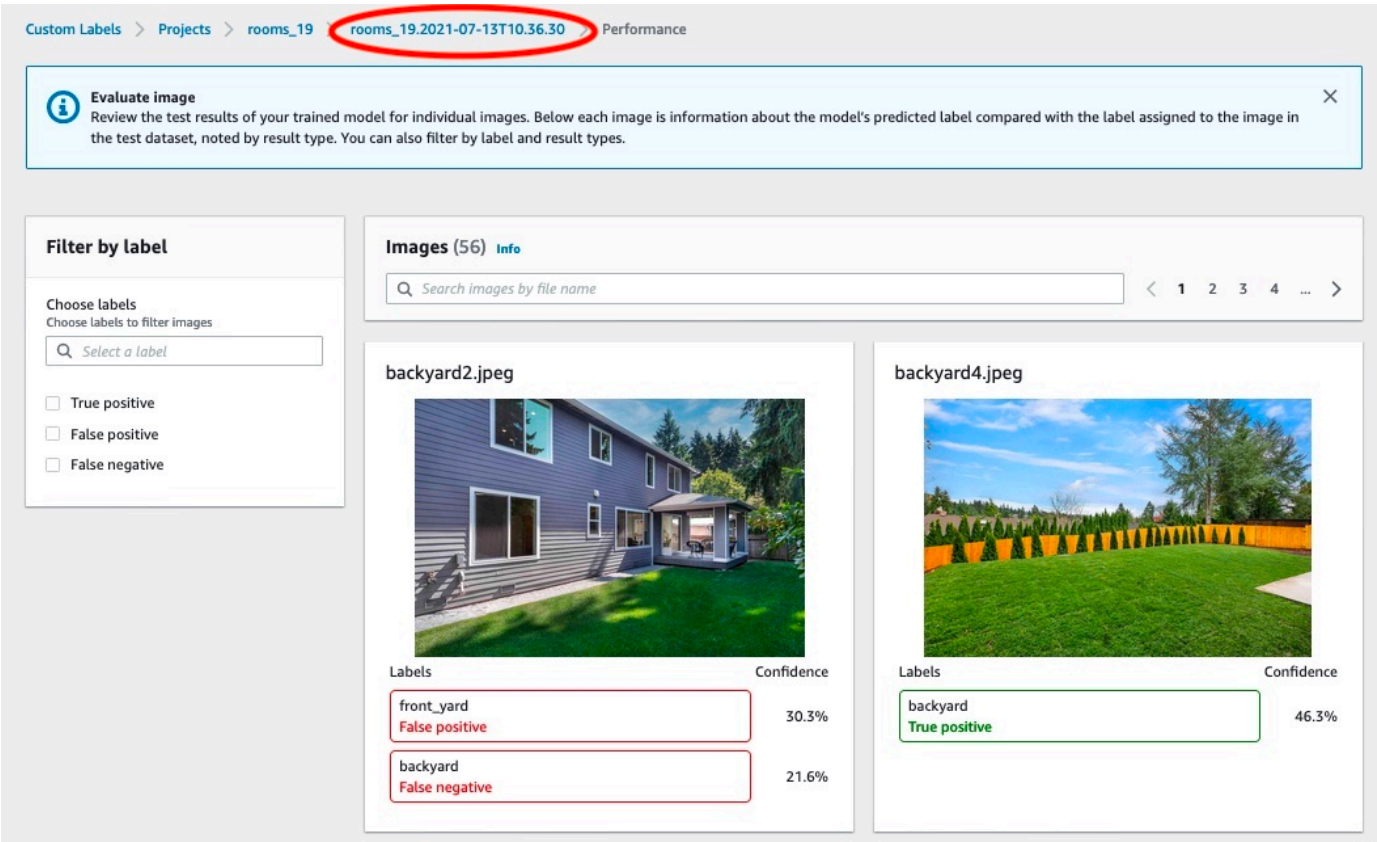


5. Pulse el botón Evaluar para ver los resultados de la evaluación. Para obtener información sobre la evaluación de un modelo, consulte Mejora de un modelo de Etiquetas personalizadas de Amazon Rekognition.

6. Seleccione Ver resultados de pruebas para ver los resultados de cada una de las imágenes de prueba. Como se ve en la siguiente captura de pantalla, el panel de evaluación muestra métricas como la puntuación F1, la precisión y la recuperación de cada etiqueta, junto con el número de imágenes de prueba. También se muestran las métricas globales, como el promedio, la precisión y la recuperación.



7. Tras ver los resultados de las pruebas, elija el nombre del modelo para volver a la página del modelo. En la siguiente captura de pantalla del panel de rendimiento, puede hacer clic para volver a la página del modelo.



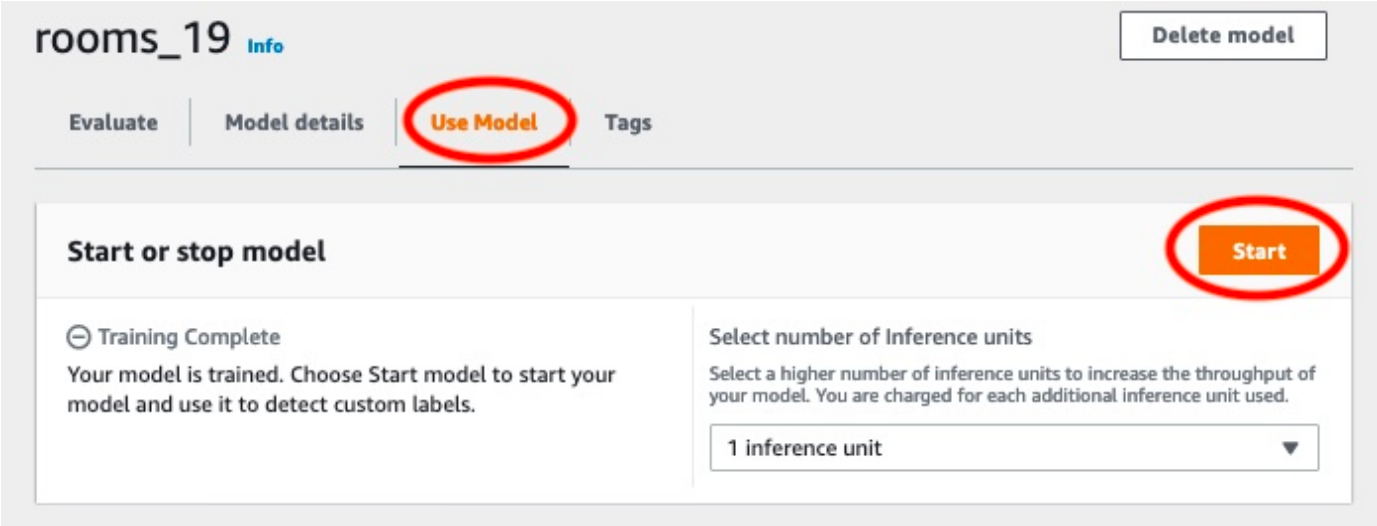
Paso 3: Ejecutar el modelo

En este paso iniciará el modelo. Después de iniciar el modelo, puede usarlo para analizar nuevas imágenes.

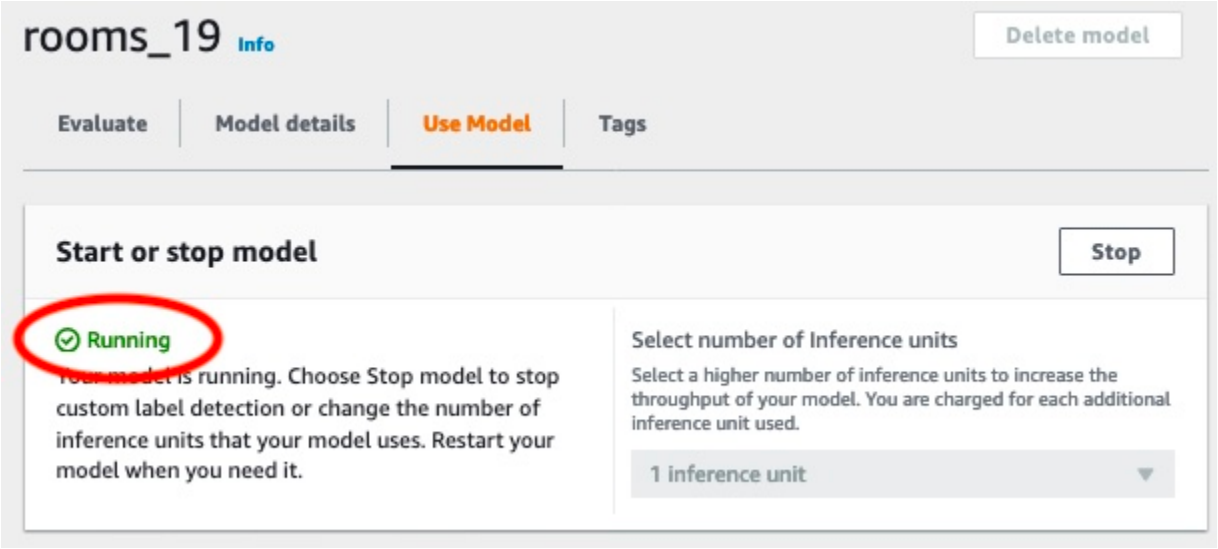
Se le cobrará por la cantidad de tiempo de ejecución del modelo. Detenga el modelo si no necesita analizar imágenes. Puede reiniciar el modelo más adelante. Para obtener más información, consulte Ejecución de un modelo de Etiquetas personalizadas de Amazon Rekognition.

Cómo iniciar el modelo

1. Seleccione la pestaña Usar modelo en la página del modelo.
2. En la sección Iniciar o detener modelo, haga lo siguiente:
   a. Elija Iniciar.
   b. En el cuadro de diálogo Iniciar modelo, seleccione Iniciar. En la imagen siguiente, se muestra el botón de inicio del panel de control del modelo.



3. Espere a que se ejecute el modelo. La siguiente captura de pantalla muestra la consola con el modelo ejecutándose y con el estado Ejecutándose en la sección Iniciar o detener el modelo.



4. Utilice su modelo para clasificar las imágenes. Para obtener más información, consulte Paso 4: Analizar una imagen con su modelo.

Paso 4: Analizar una imagen con su modelo

Para analizar una imagen, llame a la API. DetectCustomLabels En este paso, utilizas el comando `detect-custom-labels` AWS Command Line Interface (AWS CLI) para analizar una imagen de ejemplo. El AWS CLI comando se obtiene de la consola Amazon Rekognition Custom Labels. La consola configura el AWS CLI comando para usar su modelo. Solo necesita facilitar una imagen

que esté almacenada en un bucket de Amazon S3. En este tema se incluye una imagen que puede utilizar para cada proyecto de ejemplo.

> **Note**
>
> La consola también ofrece un código de ejemplo de Python.

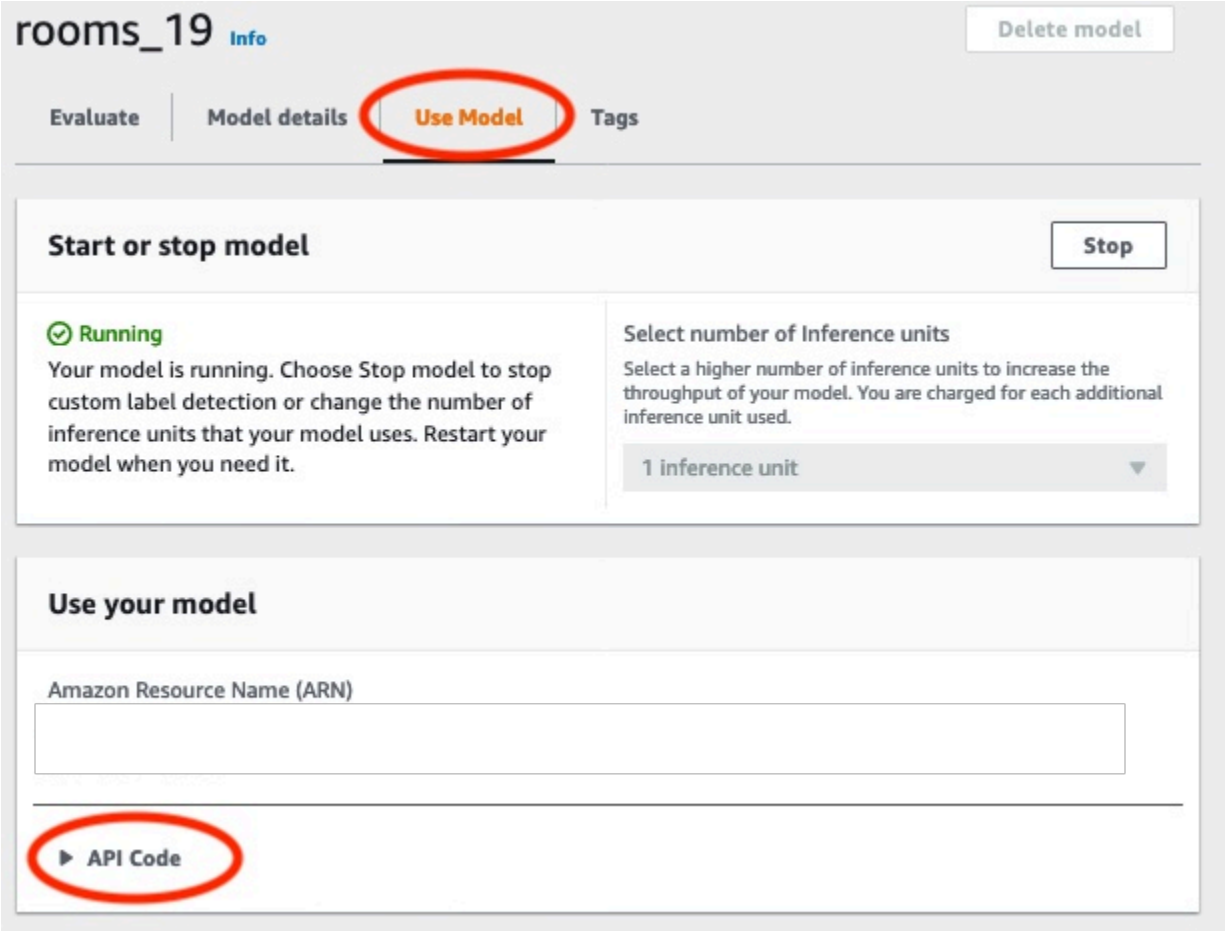
El resultado de `detect-custom-labels` incluye una lista de las etiquetas que se encuentran en la imagen, cuadros delimitadores (si el modelo encuentra ubicaciones de objetos) y la confianza que el modelo tiene en la precisión de las predicciones.

Para obtener más información, consulte Análisis de una imagen con un modelo entrenado.

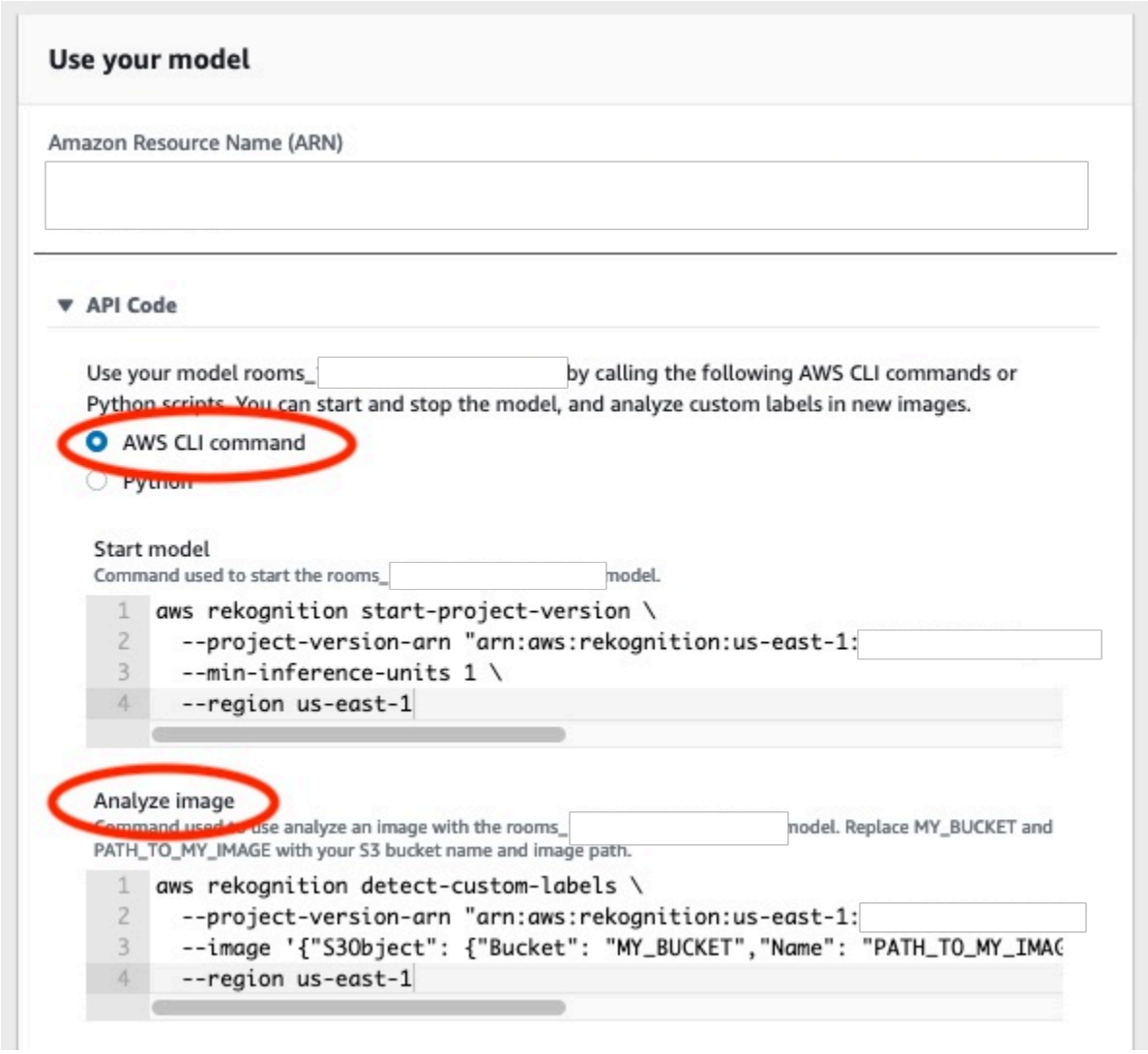
Cómo analizar una imagen (consola)

1. <textobject><phrase>Modelo con el estado Ejecutándose y el botón Detener para interrumpir la ejecución.</phrase></textobject>

   Si aún no lo ha hecho, configure. AWS CLI Para obtener instrucciones, consulte the section called “Paso 4: Configure y AWS CLIAWS SDKs”.
2. Si aún no lo ha hecho, comience a ejecutar el modelo. Para obtener más información, consulte Paso 3: Ejecutar el modelo.
3. Seleccione la pestaña Usar modelo y luego elija Código de API. El panel de estado del modelo que se muestra a continuación muestra el modelo Ejecutándose con el botón Detener para interrumpir su ejecución y una opción para mostrar la API.



4. Elija Comando AWS CLI.
5. En la sección Analizar imagen, copia el AWS CLI comando que llama`detect-custom-labels`. La siguiente imagen de la consola Rekognition muestra la sección “Analizar imagen” con el comando AWS CLI para detectar etiquetas personalizadas en una imagen mediante un modelo de machine learning e instrucciones para iniciar el modelo y proporcionar detalles de la imagen.



6. Cargue el archivo de imagen en un bucket de Amazon S3. Para obtener instrucciones, consulte Obtener una imagen de ejemplo.
7. En la línea de comandos, escriba el AWS CLI comando que copió en el paso anterior. Debería ser similar al ejemplo siguiente.

   El valor de `--project-version-arn` debe ser el nombre de recurso de Amazon (ARN) del modelo. El valor de `--region` debe ser la región de AWS en la que se creó el modelo.

   Cambie `MY_BUCKET` y `PATH_TO_MY_IMAGE` por el bucket de Amazon S3 y la imagen que utilizó en el paso anterior.

   Si va a usar el perfil custom-labels-access para obtener las credenciales, añada el parámetro `--profile custom-labels-access`.

```
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
```

Si el modelo encuentra objetos, escenas y conceptos, el resultado JSON del comando AWS CLI debería tener un aspecto similar al siguiente. `Name` es el nombre de la etiqueta de imagen que detectó el modelo. `Confidence` (0-100) es la confianza del modelo en la precisión de la predicción.

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

Si el modelo encuentra la ubicación de los objetos o encuentra la marca, aparecerán los cuadros delimitadores etiquetados. `BoundingBox` contiene la ubicación de un cuadro que rodea el objeto. `Name` es el objeto que el modelo encontró en el cuadro delimitador. `Confidence` es la confianza del modelo en que el cuadro delimitador contenga el objeto.

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

```
}
```

8. Siga utilizando el modelo para analizar otras imágenes. Detenga el modelo si deja de usarlo. Para obtener más información, consulte Paso 5: Detener el modelo.

Obtener una imagen de ejemplo

Puede utilizar las siguientes imágenes para la operación `DetectCustomLabels`. Hay una imagen para cada proyecto. Para utilizar las imágenes, cárguelas en un bucket de S3.

Cómo usar una imagen de ejemplo

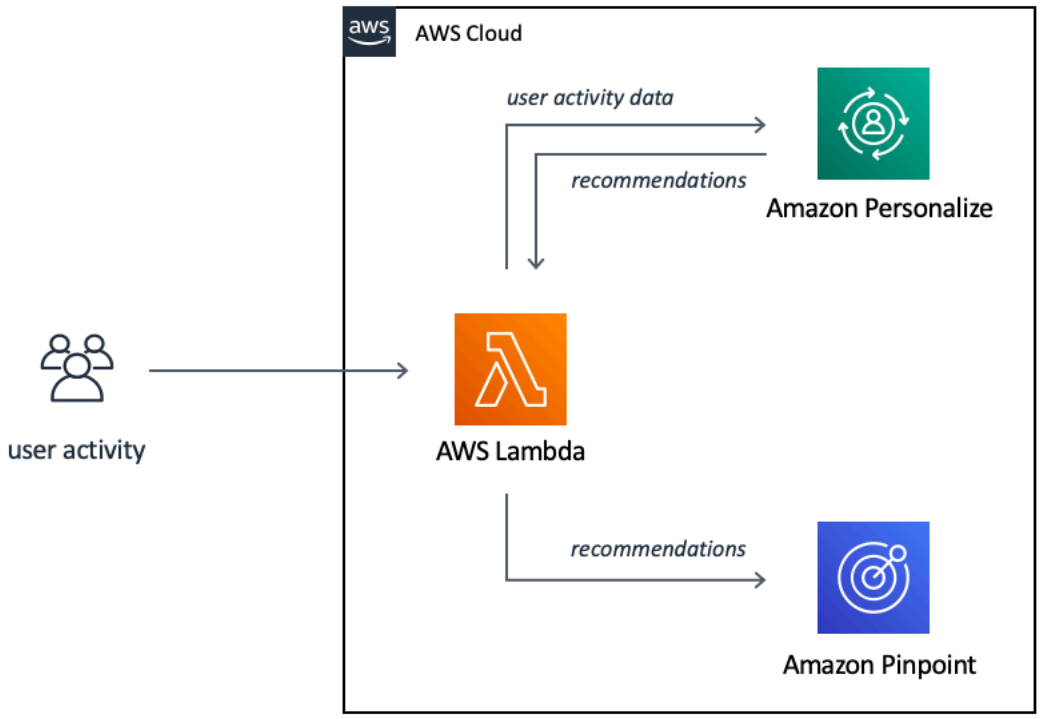
1. Haga clic con el botón derecho en la siguiente imagen que se corresponda con el proyecto de ejemplo que está utilizando. A continuación, seleccione Guardar imagen para guardar la imagen en su ordenador. La opción del menú puede ser diferente, según el navegador que utilice.
2. Sube la imagen a un bucket de Amazon S3 que sea propiedad de tu AWS cuenta y que se encuentre en la misma AWS región en la que utilizas las etiquetas personalizadas de Amazon Rekognition.

   Para ver las instrucciones, consulte Carga de objetos en Amazon S3 en la Guía del usuario de Amazon Simple Storage Service.

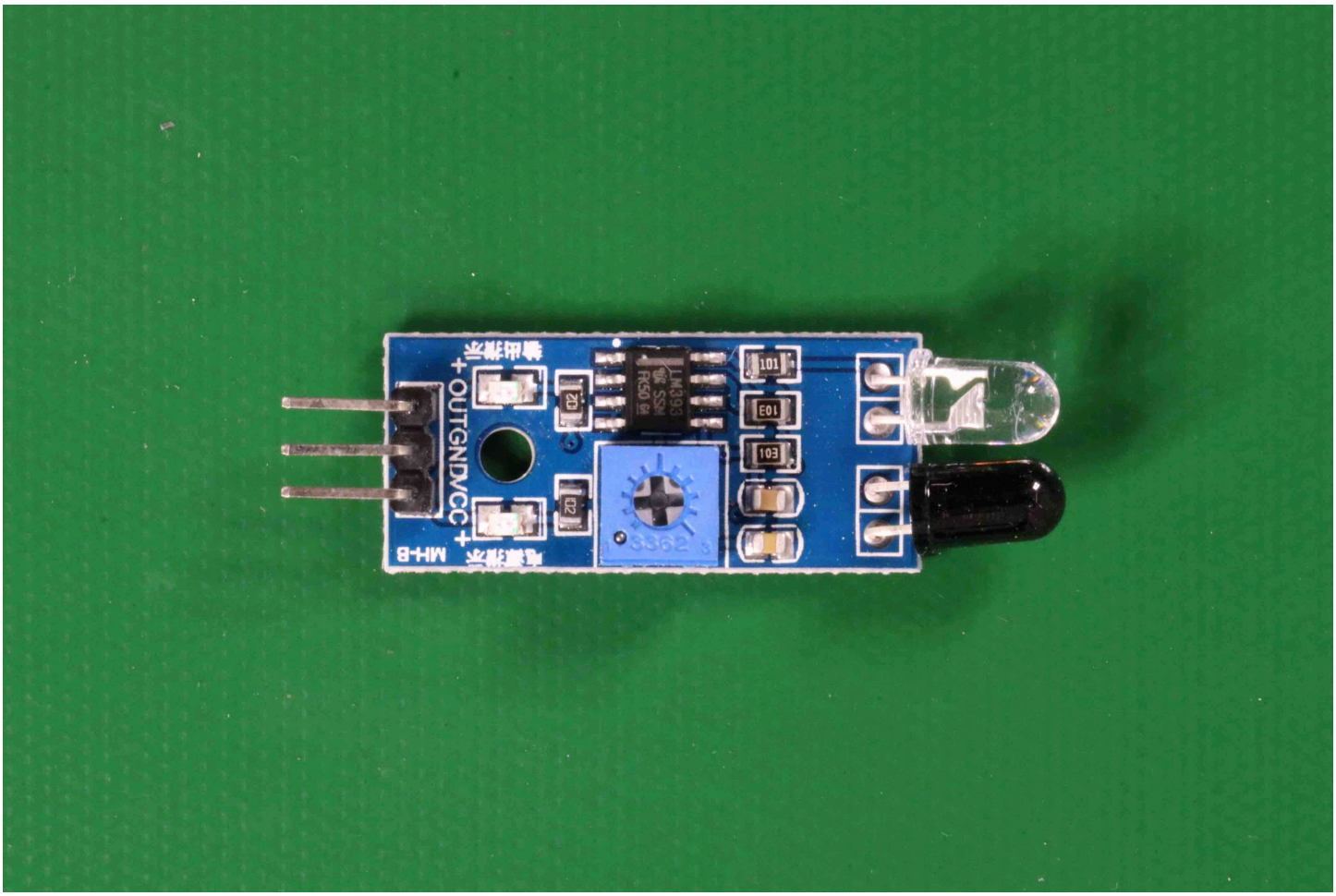
Clasificación de imágenes

Clasificación de etiquetas múltiples

Detección de marcas



Localización de objetos

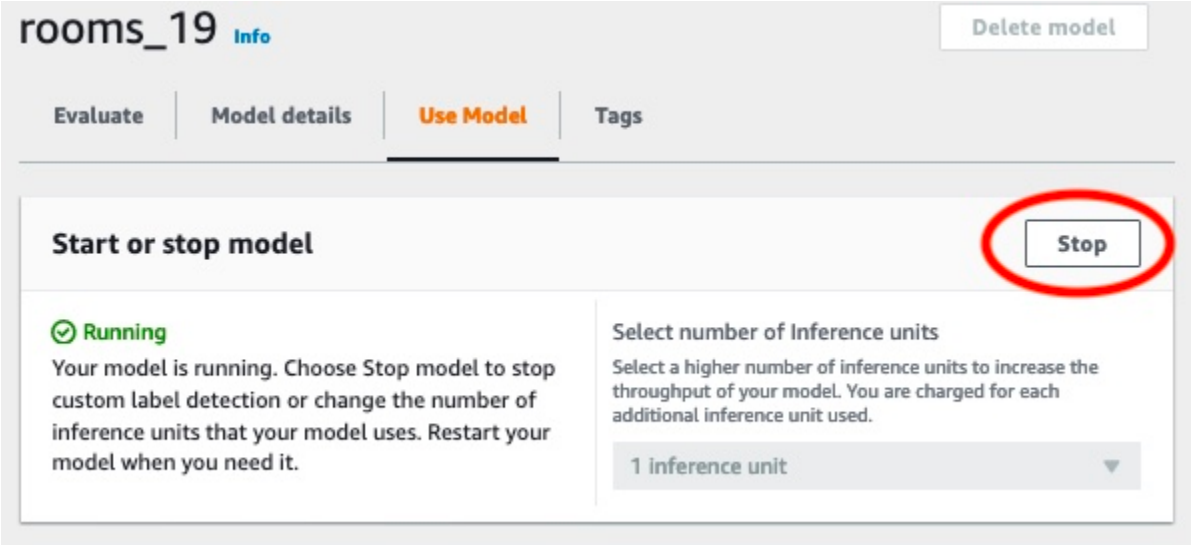


Paso 5: Detener el modelo

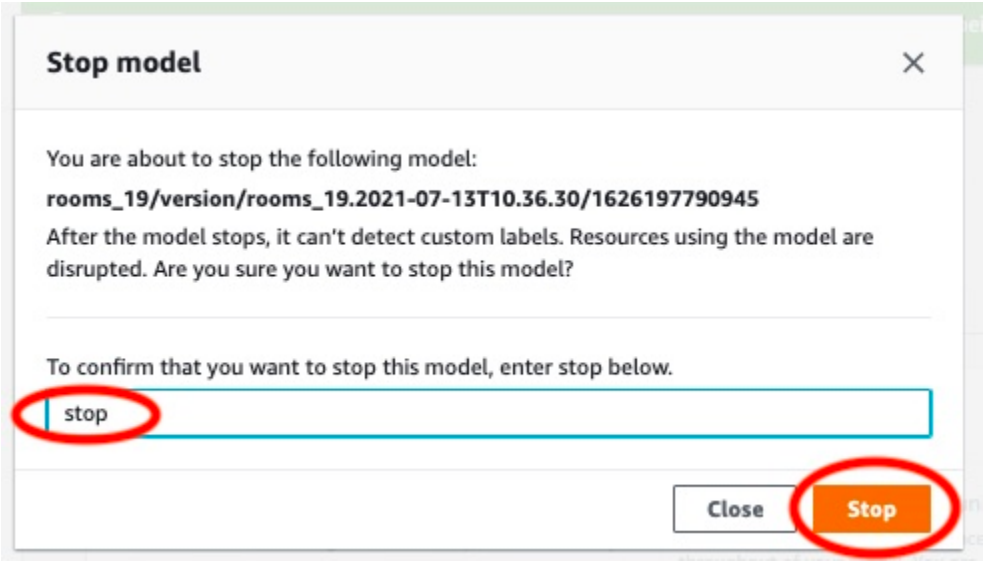
En este paso, dejará de ejecutar el modelo. Se le cobrará por la cantidad de tiempo de ejecución del modelo. Si ha terminado de usar el modelo, debe detenerlo.

Cómo detener su modelo

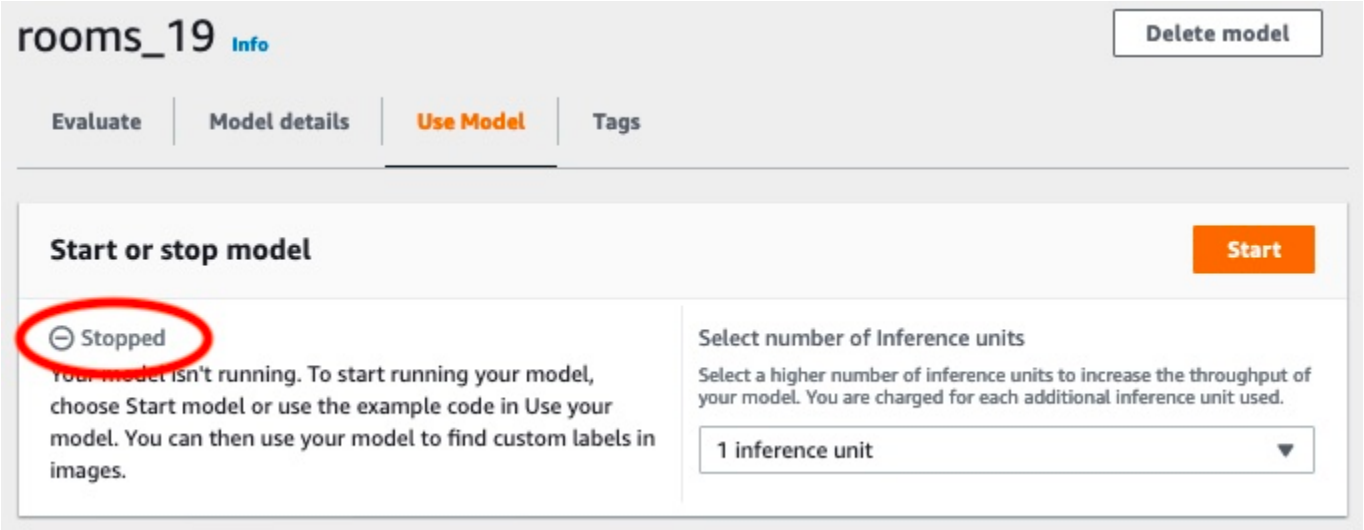
1. En la sección Iniciar o detener modelo, seleccione Detener.



2. En el cuadro de diálogo Detener modelo, escriba detener para confirmar que desea detener el modelo.



3. Seleccione Detener para detener el modelo. El modelo se habrá detenido cuando el estado de la sección Iniciar o detener modelo tenga la opción Detenido activada. En la siguiente captura de pantalla, la sección de la interfaz de usuario tiene la opción de iniciar o detener un modelo de machine learning. Estado de modelo “Detenido” con el botón “Iniciar” para iniciar el modelo y un menú desplegable para seleccionar el número de unidades de inferencia.



Paso 6: Siguientes pasos

Una vez que haya terminado de probar los proyectos de ejemplo, puede usar sus propias imágenes y conjuntos de datos para crear su propio modelo. Para obtener más información, consulte Qué es Etiquetas personalizadas de Amazon Rekognition.

Use la información de etiquetado de la siguiente tabla para entrenar modelos similares a los de los proyectos de ejemplo.

Ejemplo	Imágenes de entrenamiento	Imágenes de prueba
Clasificación de imágenes (habitaciones)	1 etiqueta de imagen por imagen	1 etiqueta de imagen por imagen
Clasificación de etiquetas múltiples (flores)	Etiquetas múltiples de imagen por imagen	Etiquetas múltiples de imagen por imagen
Detección de marcas (logotipos)	Etiquetas de imagen (también puede utilizar cuadros delimitadores etiquetados)	Cuadros delimitadores etiquetados
Localización de imágenes (placas de circuitos)	Cuadros delimitadores etiquetados	Cuadros delimitadores etiquetados

En Clasificación de imágenes se explica cómo crear un proyecto, conjuntos de datos y modelos para un modelo de clasificación de imágenes.

Para obtener información detallada sobre cómo crear conjuntos de datos y modelos de entrenamiento, consulte Creación de un modelo de Etiquetas personalizadas de Amazon Rekognition.

Clasificación de imágenes

En este tutorial, se explica cómo crear el proyecto y los conjuntos de datos para un modelo que clasifica los objetos, las escenas y los conceptos que se encuentran en una imagen. El modelo clasifica toda la imagen. Por ejemplo, según este tutorial, puede entrenar un modelo para que reconozca las ubicaciones de una casa, como la sala de estar o la cocina. El tutorial también le enseña a utilizar el modelo para analizar imágenes.

Antes de comenzar el tutorial, le recomendamos que lea Qué es Etiquetas personalizadas de Amazon Rekognition.

En este tutorial, creará los conjuntos de datos de entrenamiento y de prueba cargando imágenes en su ordenador local. Más adelante, asignará etiquetas de imagen a las imágenes de los conjuntos de datos de entrenamiento y prueba.

El modelo que cree clasificará las imágenes como pertenecientes al grupo de etiquetas de imagen que asigne a las imágenes del conjunto de datos de entrenamiento. Por ejemplo, si el conjunto de etiquetas de imagen del conjunto de datos de entrenamiento es `kitchen`, `living_room`, `patio` y `backyard`, el modelo podrá encontrar todas esas etiquetas de imagen en una sola imagen.

> **Note**
>
> Puede crear modelos para distintos fines, como encontrar la ubicación de objetos en una imagen. Para obtener más información, consulte Cómo decidir el tipo de modelo.

Paso 1: Reunir las imágenes

Necesitará dos grupos de imágenes. Un grupo para agregar a su conjunto de datos de entrenamiento. Y otro grupo para agregar a su conjunto de datos de prueba. Las imágenes deben representar los objetos, las escenas y los conceptos que desee que clasifique su modelo. Las imágenes deben tener formato PNG o JPEG. Para obtener más información, consulte Preparación de imágenes.

Debe tener al menos 10 imágenes para su conjunto de datos de entrenamiento y 10 imágenes para su conjunto de datos de prueba.

Si aún no tiene ninguna imagen, use las de ejemplo del proyecto de clasificación Habitaciones. Tras crear el proyecto, las imágenes de entrenamiento y de prueba estarán disponibles en las siguientes ubicaciones del bucket de Amazon S3:

- Imágenes de entrenamiento: `s3://custom-labels-console-`*`region`*`-`*`numbers`*`/assets/rooms_`*`version number`*`_test_dataset/`
- Imágenes de prueba: `s3://custom-labels-console-`*`region`*`-`*`numbers`*`/assets/rooms_`*`version number`*`_test_dataset/`

`regiones` la AWS región en la que utiliza la consola de etiquetas personalizadas de Amazon Rekognition. `numberses` un valor que la consola asigna al nombre del bucket. `Version numberes` el número de versión del proyecto de ejemplo, empezando por 1.

El siguiente procedimiento almacena las imágenes del proyecto Habitaciones en carpetas locales del equipo denominadas `training` y `test`.

Cómo descargar los archivos de imagen del proyecto Habitaciones

1. Cree el proyecto Habitaciones. Para obtener más información, consulte Paso 1: Elegir un proyecto de ejemplo.
2. Abra el símbolo del sistema y escriba el siguiente comando para descargar las imágenes de entrenamiento.

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

3. En el símbolo del sistema, introduzca el siguiente comando para descargar las imágenes de prueba.

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

4. Mueva dos de las imágenes de la carpeta de entrenamiento a una carpeta independiente que elija. Usará las imágenes para probar el modelo entrenado en Paso 9: Analizar una imagen con su modelo.

Paso 2: Decidir las clases

Haga una lista de las clases que desee que busque su modelo. Por ejemplo, si va a entrenar un modelo para que reconozca las habitaciones de una casa, puede clasificar la siguiente imagen como `living_room`.

Cada clase se asigna a una etiqueta de imagen. Más adelante, asignará las etiquetas de imagen a las imágenes de los conjuntos de datos de entrenamiento y de prueba.

Si va a usar las imágenes del proyecto de ejemplo Habitaciones, las etiquetas de imagen son terraza, baño, dormitorio, armario, camino_entrada, plano_planta, patio_delantero, cocina, sale_estar y patio.

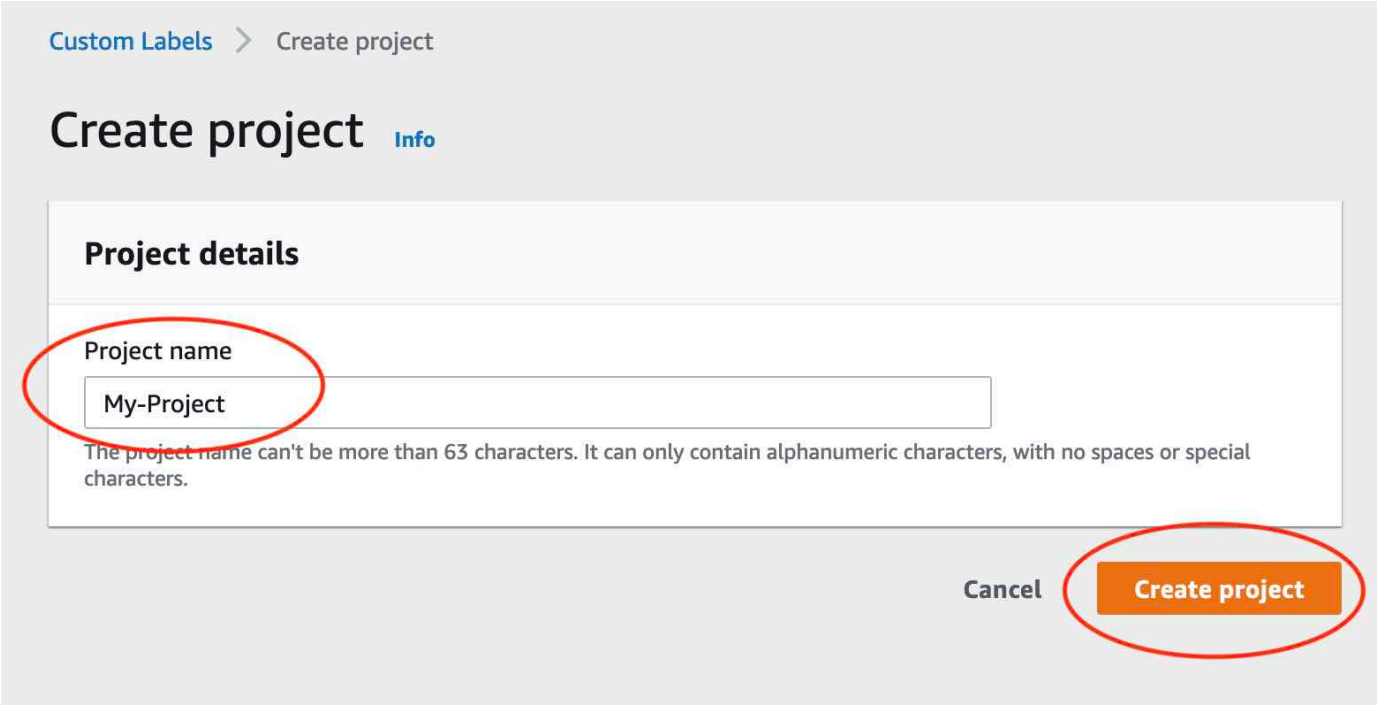
Paso 3: Crear un proyecto

Para administrar sus conjuntos de datos y modelos, debe crear un proyecto. Cada proyecto debe usarse para una aplicación concreta, como reconocer las habitaciones de una casa.

Cómo crear un proyecto (consola)

1. Si aún no lo ha hecho, configure la consola de Etiquetas personalizadas de Amazon Rekognition. Para obtener más información, consulte Configuración de Etiquetas personalizadas de Amazon Rekognition.
2. Inicie sesión en la consola Amazon Rekognition AWS Management Console y ábrala en. https://console.aws.amazon.com/rekognition/
3. En el panel izquierdo, elija Usar etiquetas personalizadas. Se abrirá la página de inicio de Etiquetas personalizadas de Amazon Rekognition.

4. En la página de inicio de Etiquetas personalizadas de Amazon Rekognition, seleccione Comenzar.
5. En el panel de navegación izquierdo, elija Proyectos.
6. En la página Proyectos, elija Crear proyecto.
7. En Nombre del proyecto, introduzca un nombre para el proyecto.
8. Seleccione Crear proyecto para crearlo.



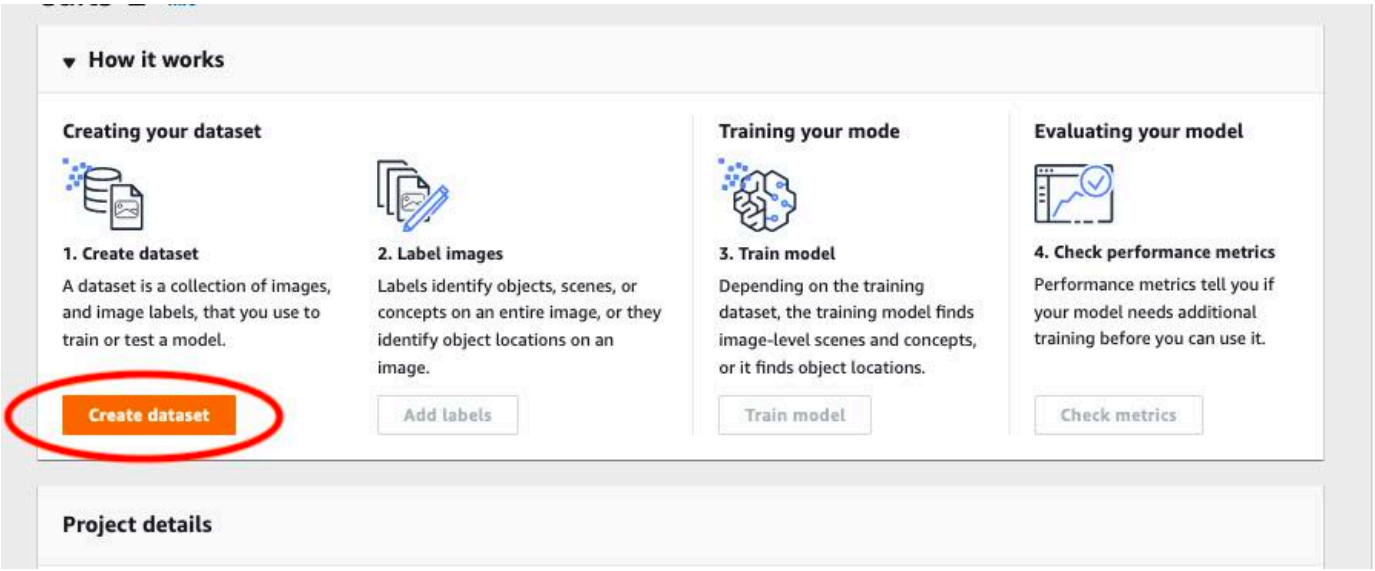
Paso 4: Crear conjuntos de datos de entrenamiento y de prueba

En este paso, creará un conjunto de datos de entrenamiento y de prueba cargando imágenes de su ordenador local. Puede cargar hasta 30 imágenes a la vez. Si tiene que cargar muchas imágenes, piense en crear los conjuntos de datos importándolos desde un bucket de Amazon S3. Para obtener más información, consulte Importar imágenes desde un bucket de Amazon S3.

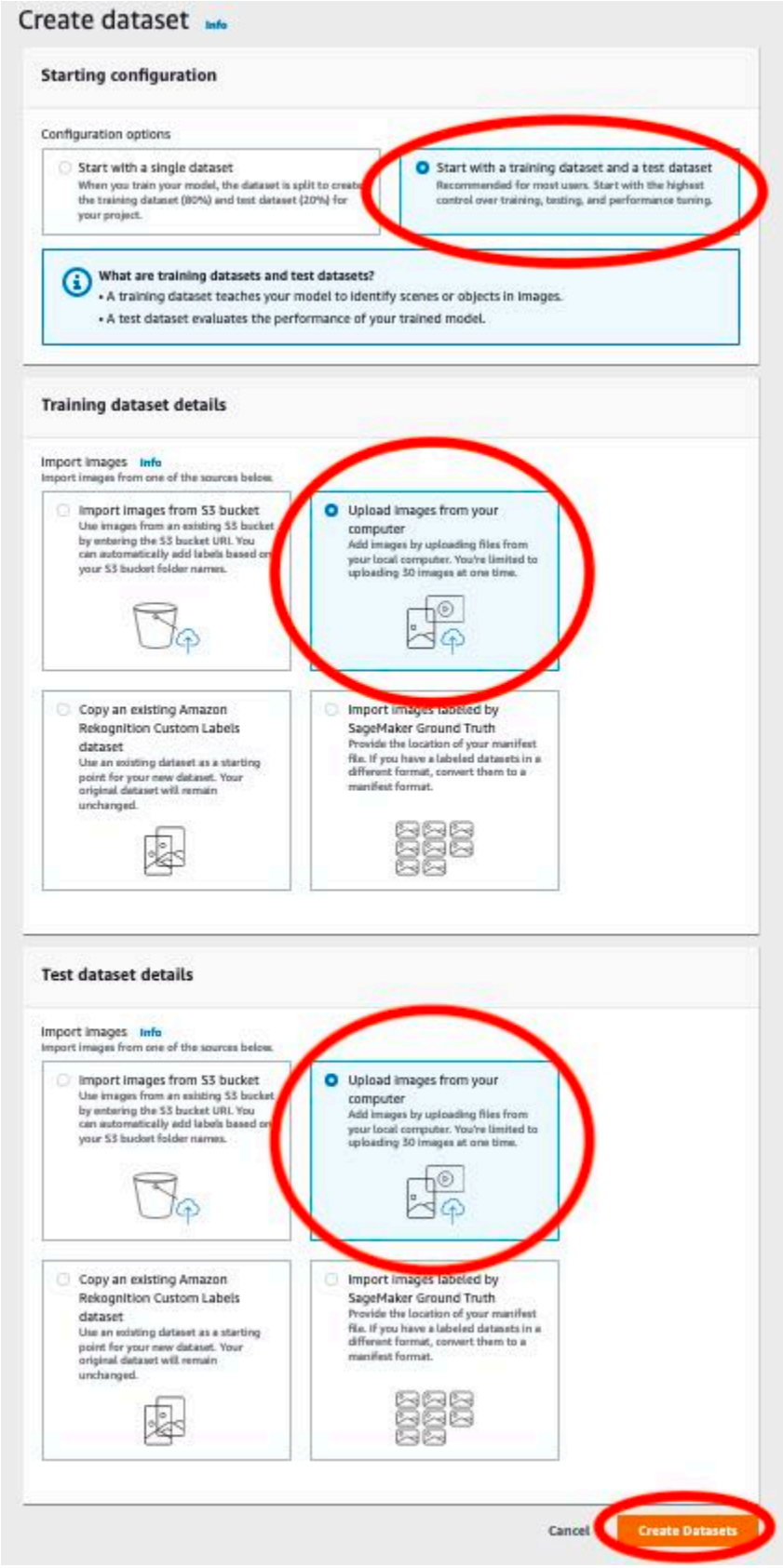
Para obtener más información sobre los conjuntos de datos, consulte Administración de conjuntos de datos.

Cómo crear un conjunto de datos con imágenes en un ordenador local (consola)

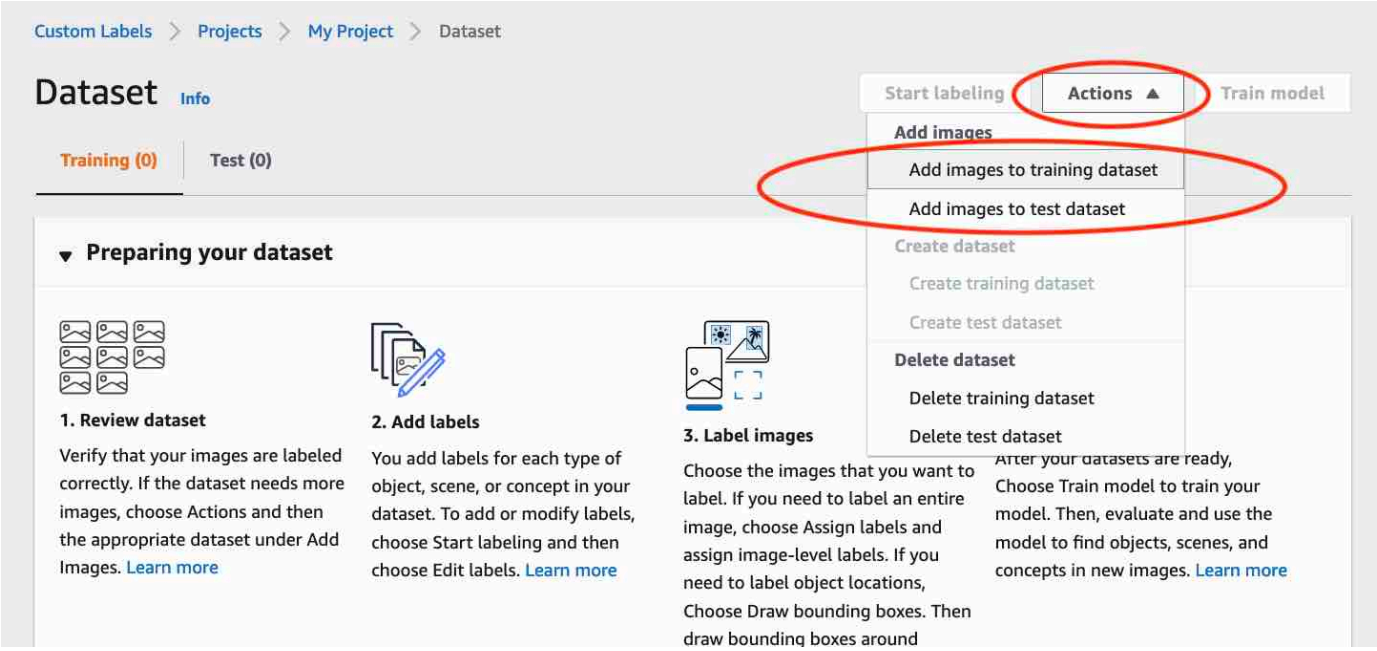
1. En la página de detalles del proyecto, elija Crear conjunto de datos.



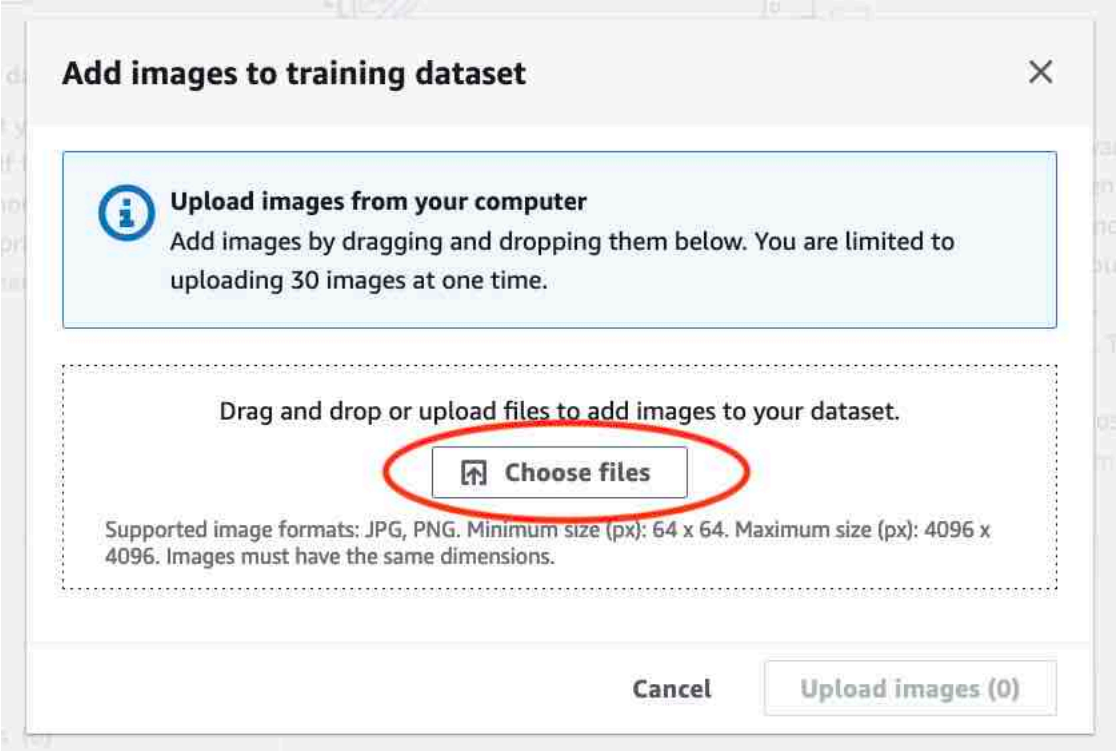
2. En la sección Configuración inicial, seleccione Comenzar con un conjunto de datos de entrenamiento y un conjunto de datos de prueba.
3. En la sección Detalles del conjunto de datos de entrenamiento, seleccione Cargar imágenes del ordenador.
4. En la sección Detalles del conjunto de datos de prueba, seleccione Cargar imágenes del ordenador.
5. Elija Crear conjuntos de datos.


Create dataset Info
Starting configuration
Configuration options
Start with a single dataset
When you train your model, the dataset is split to create the training dataset (80%) and test dataset (20%) for your project.
Start with a training dataset and a test dataset
Recommended for most users. Start with the highest control over training, testing, and performance tuning.
What are training datasets and test datasets?
• A training dataset teaches your model to identify scenes or objects in images.
• A test dataset evaluates the performance of your trained model.
Training dataset details
Import images Info
Import images from one of the sources below.
Import images from S3 bucket
Use images from an existing S3 bucket by entering the S3 bucket URL. You can automatically add labels based on your S3 bucket folder names.
Upload images from your computer
Add images by uploading files from your local computer. You're limited to uploading 30 images at one time.
Copy an existing Amazon Rekognition Custom Labels dataset
Use an existing dataset as a starting point for your new dataset. Your original dataset will remain unchanged.
Import images labeled by SageMaker Ground Truth
Provide the location of your manifest file. If you have a labeled datasets in a different format, convert them to a manifest format.
Test dataset details
Import images Info
Import images from one of the sources below.
Import images from S3 bucket
Use images from an existing S3 bucket by entering the S3 bucket URL. You can automatically add labels based on your S3 bucket folder names.
Upload images from your computer
Add images by uploading files from your local computer. You're limited to uploading 30 images at one time.
Copy an existing Amazon Rekognition Custom Labels dataset
Use an existing dataset as a starting point for your new dataset. Your original dataset will remain unchanged.
Import images labeled by SageMaker Ground Truth
Provide the location of your manifest file. If you have a labeled datasets in a different format, convert them to a manifest format.
Cancel
Create Datasets

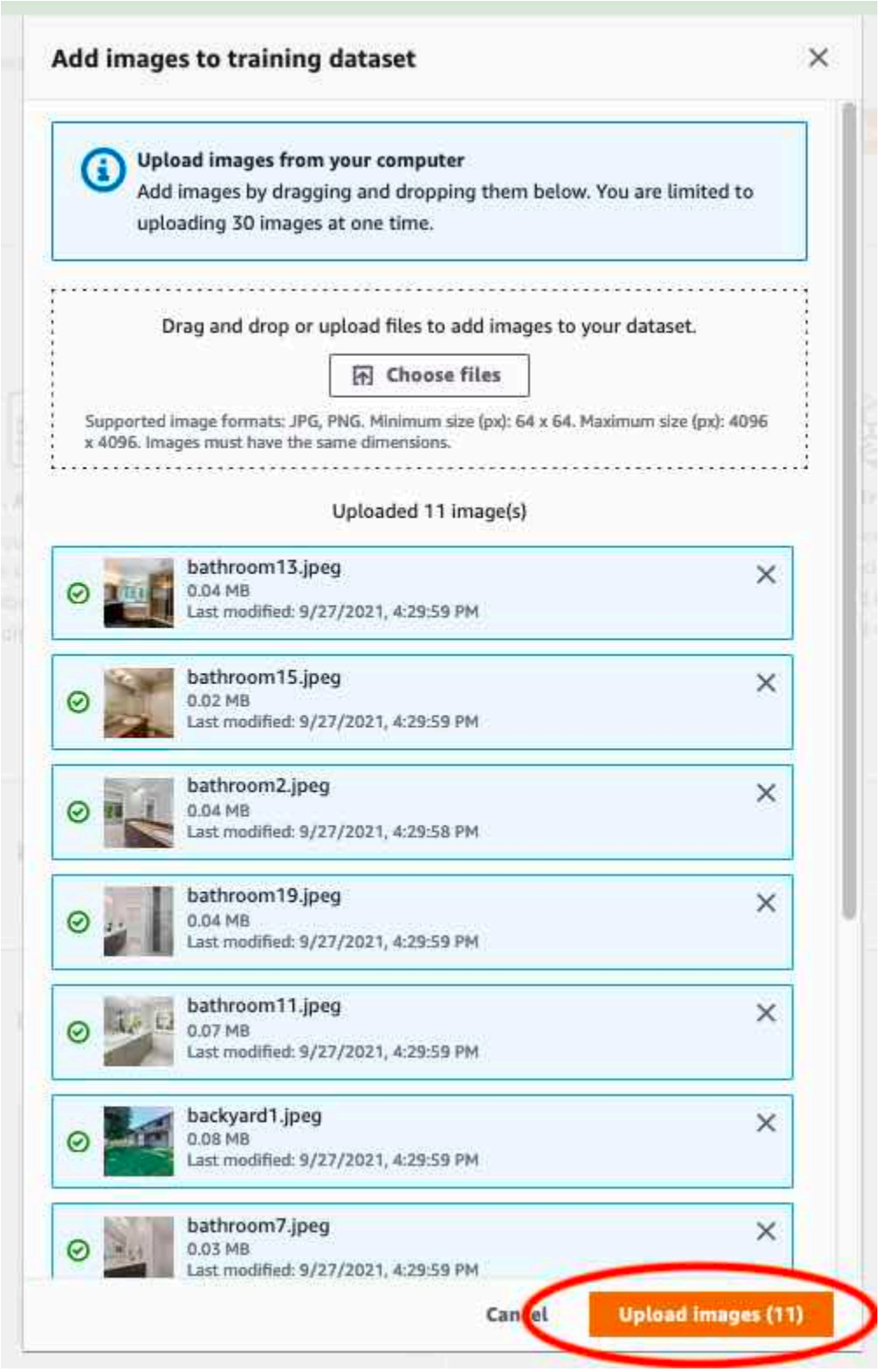
6. Se abrirá la página de conjuntos de datos del proyecto con las pestañas Entrenamiento y Prueba con los conjuntos de datos respectivos.
7. En la página del conjunto de datos, seleccione la pestaña Entrenamiento.
8. Seleccione Acciones y luego Agregar imágenes al conjunto de datos de entrenamiento.



9. En el cuadro de diálogo Agregar imágenes al conjunto de datos de entrenamiento, seleccione Elegir archivos.



10. Escoja las imágenes que quiera cargar en el conjunto de datos. Puede cargar hasta 30 imágenes a la vez.
11. Seleccione Cargar imágenes. Etiquetas personalizadas de Amazon Rekognition puede tardar unos segundos en agregar las imágenes al conjunto de datos.



12. Si tiene más imágenes que agregar al conjunto de datos de entrenamiento, repita los pasos del 9 al 12.
13. Elija la pestaña Prueba.

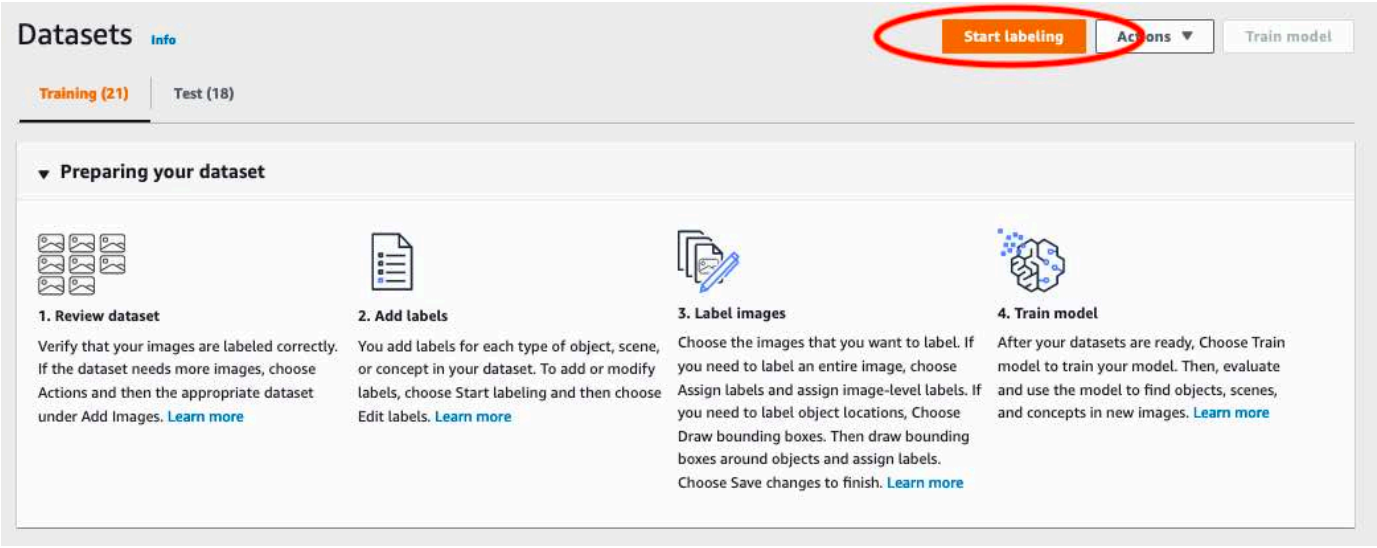
14. Repita los pasos del 8 al 12 para agregar imágenes al conjunto de datos de prueba. En el paso 8, seleccione Acciones y luego Agregar imágenes al conjunto de datos de prueba.

Paso 5: Agregar etiquetas al proyecto

En este paso, agregue una etiqueta al proyecto para cada una de las clases que identificó en el paso Paso 2: Decidir las clases.

Cómo agregar una nueva etiqueta (consola)

1. En la página de la galería de conjuntos de datos, seleccione Empezar a etiquetar para activar el modo de etiquetado.



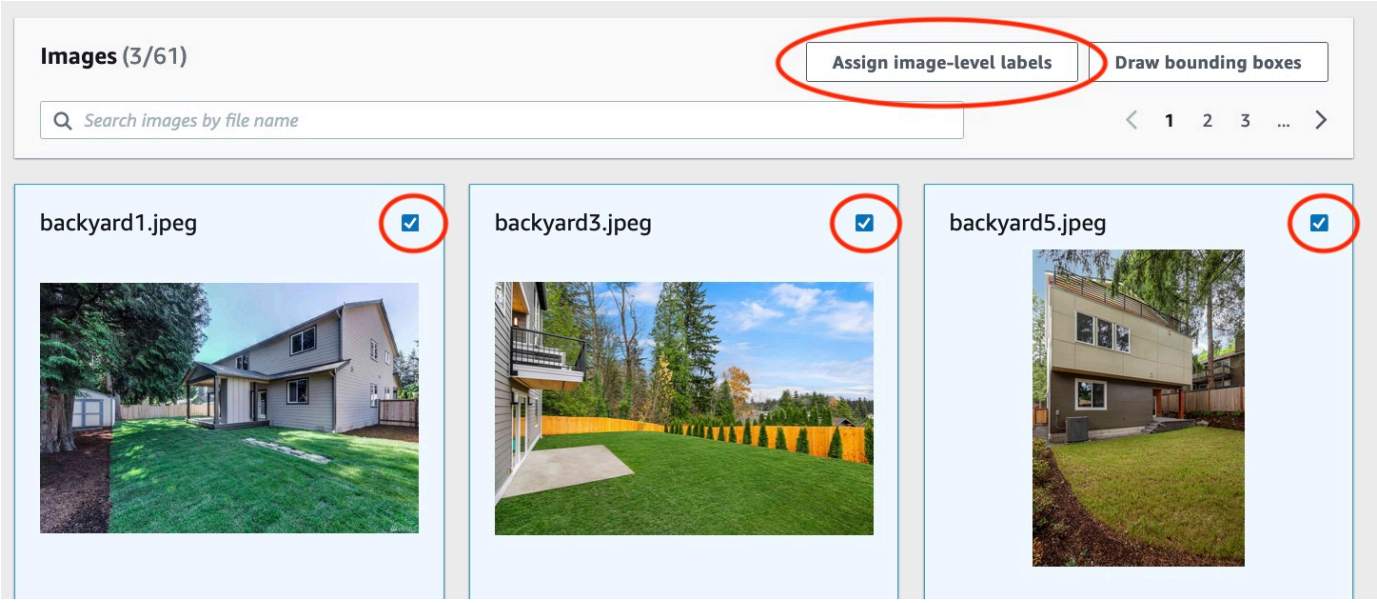
2. En la sección Etiquetas de la galería de conjuntos de datos, elija Editar etiquetas para abrir el cuadro de diálogo Administrar etiquetas.
3. En el cuadro de edición, introduzca un nombre de etiqueta nuevo.
4. Elija Agregar etiqueta.
5. Repita los pasos 3 y 4 hasta terminar de crear todas las etiquetas que necesite.
6. Elija Guardar para guardar las etiquetas que haya añadido.

Paso 6: Asignar etiquetas de imagen a los conjuntos de datos de entrenamiento y de prueba

En este paso, asignará una sola clase de imagen a cada imagen en los conjuntos de datos de entrenamiento y de prueba. La etiqueta de imagen es la clase que representa cada imagen.

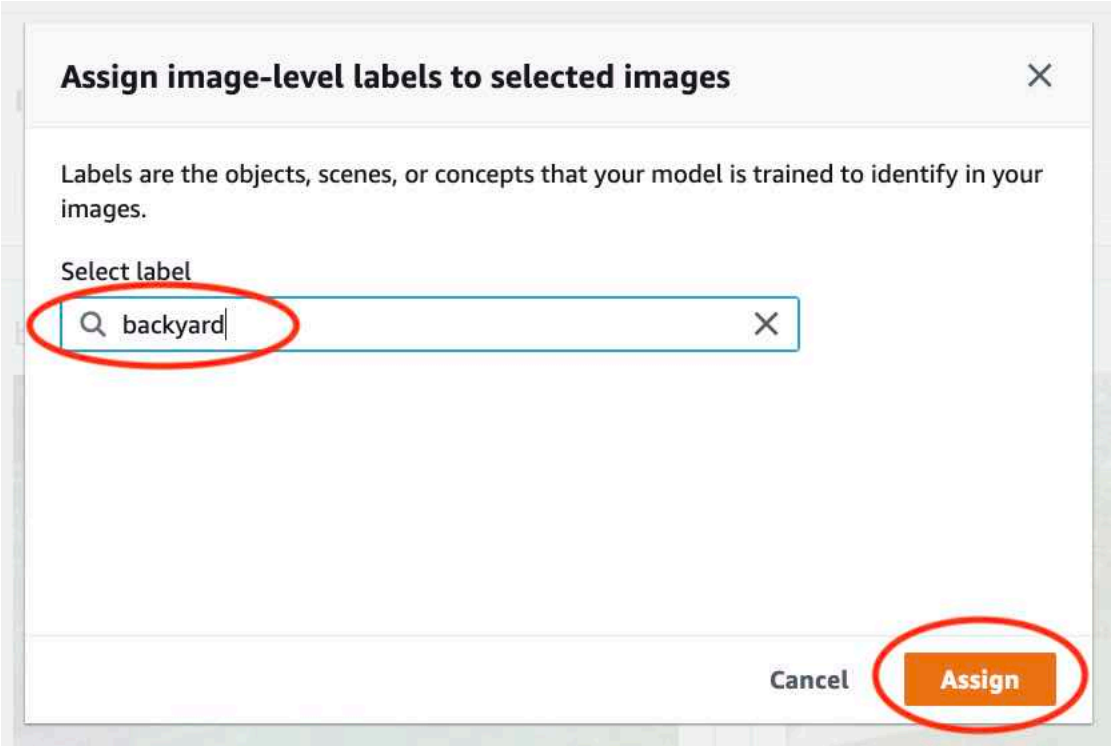
Cómo asignar etiquetas de imagen a una imagen (consola)

1. En la página Conjuntos de datos, elija la pestaña Entrenamiento.
2. Elija Comenzar a etiquetar para activar el modo de etiquetado.
3. Seleccione una o varias imágenes a las que desee agregar etiquetas. Solo puede seleccionar imágenes en una sola página a la vez. Cómo seleccionar una serie contigua de imágenes en una página:
   a. Seleccione la primera imagen.
   b. Deje pulsada la tecla Mayús.
   c. Seleccione la segunda imagen. Se seleccionarán también las imágenes entre la primera y la segunda imagen.
   d. Suelte la tecla Mayús.
4. Elija Asignar etiquetas de imagen.



5. En el cuadro de diálogo Asignar etiquetas de imagen a las imágenes seleccionadas, seleccione la etiqueta que desee asignar a la imagen o imágenes.

6. Elija Asignar para asignar una etiqueta a la imagen.



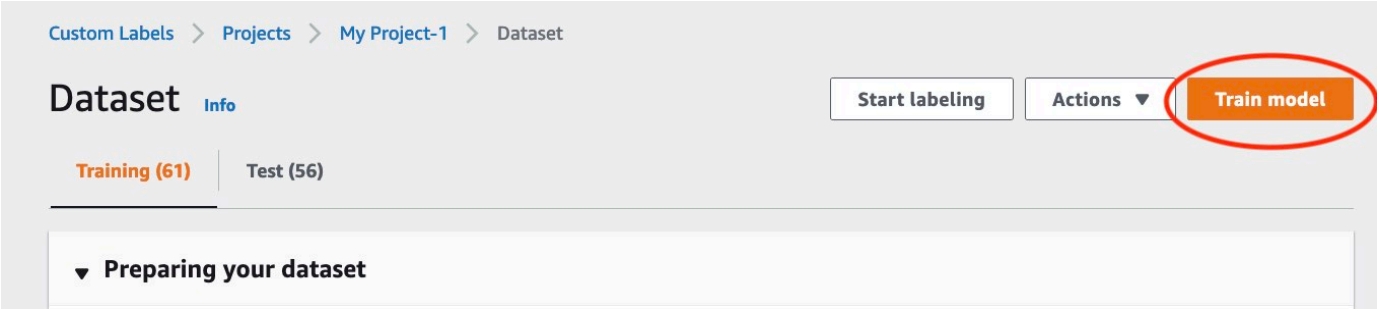
7. Repita el proceso de etiquetado hasta que todas las imágenes estén anotadas con las etiquetas necesarias.
8. Elija la pestaña Prueba.
9. Repita los pasos para asignar etiquetas de imagen a las imágenes del conjunto de datos de prueba.

Paso 7: Entrenar un modelo

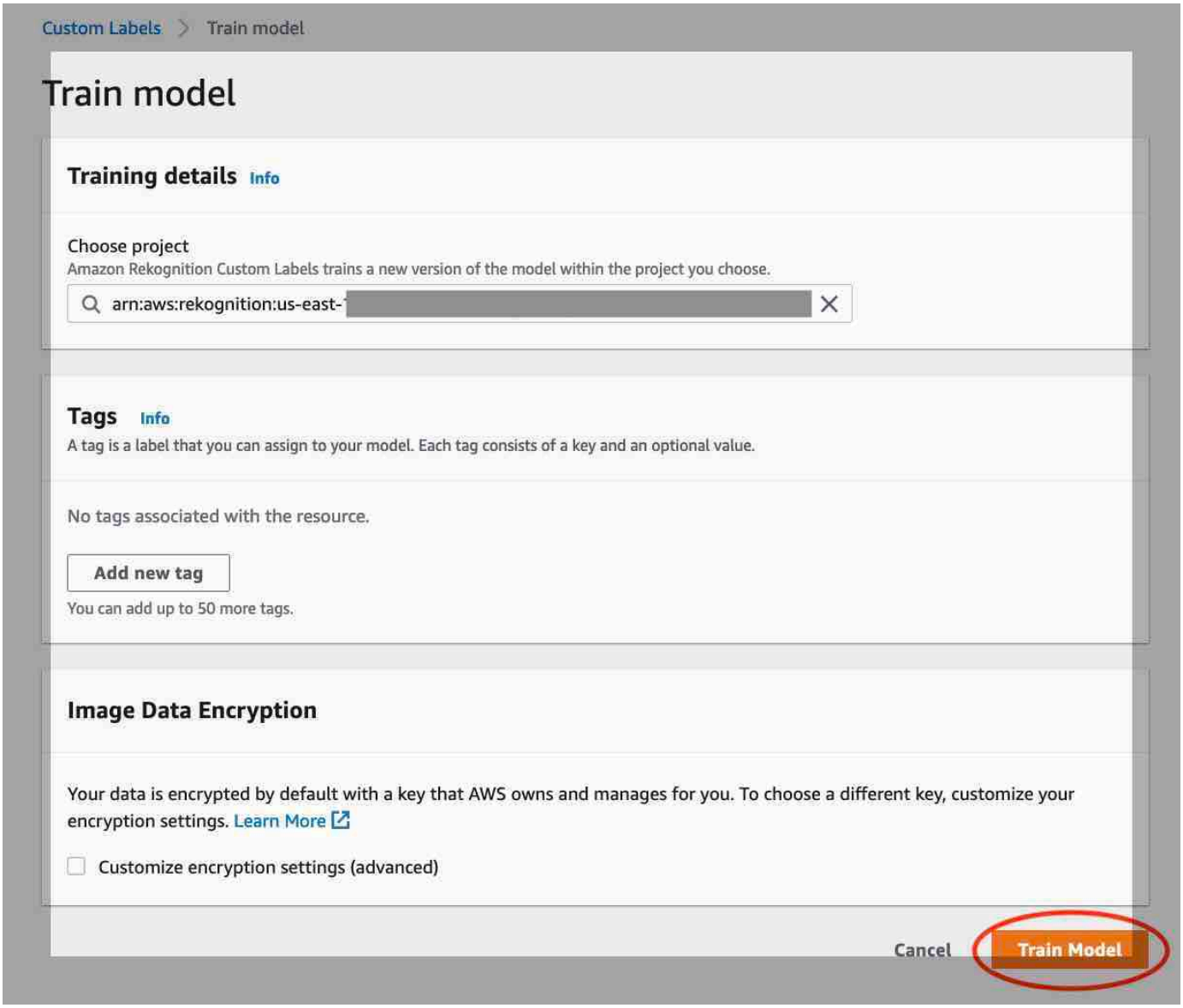
Siga los pasos que se indican a continuación para entrenar el modelo. Para obtener más información, consulte Entrenamiento de un modelo de Etiquetas personalizadas de Amazon Rekognition.

Cómo entrenar su modelo (consola)

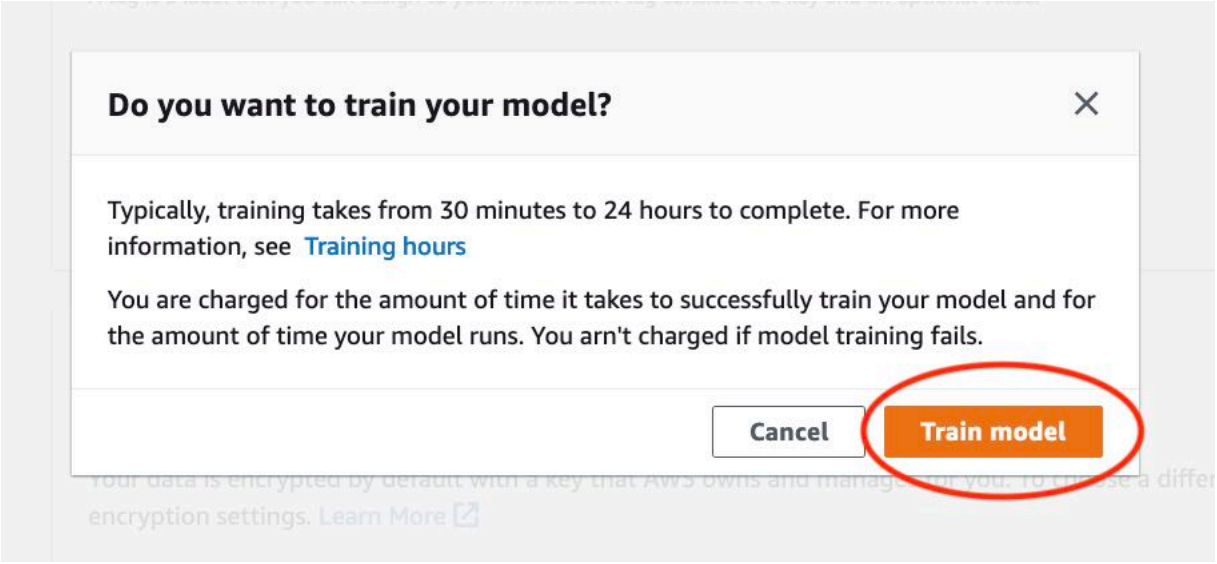
1. En la página Conjunto de datos, seleccione Entrenar modelo.



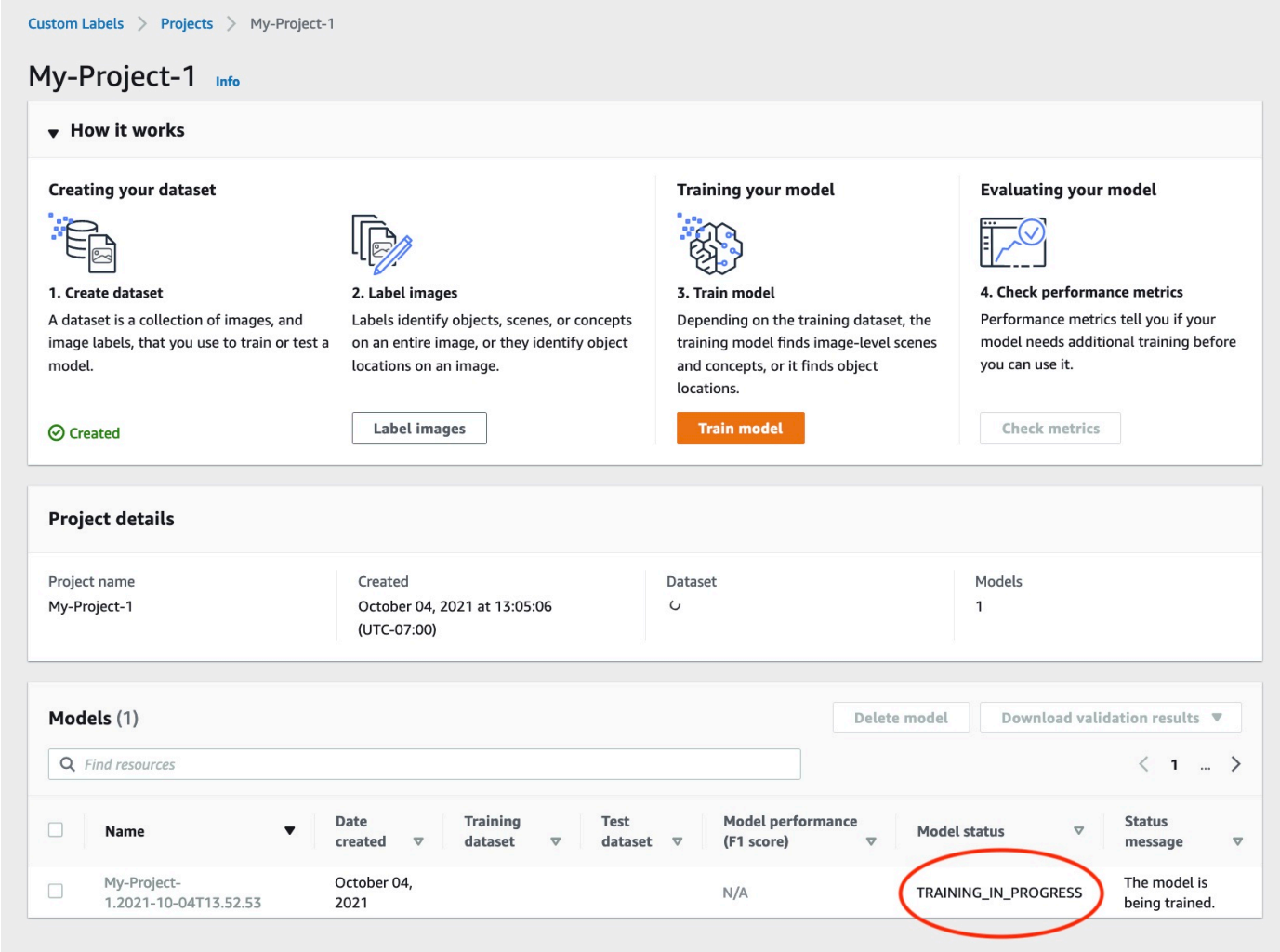
2. En la página Entrenar modelo, elija Entrenar modelo. El nombre de recurso de Amazon (ARN) del proyecto se encuentra en el cuadro de edición Elegir proyecto.



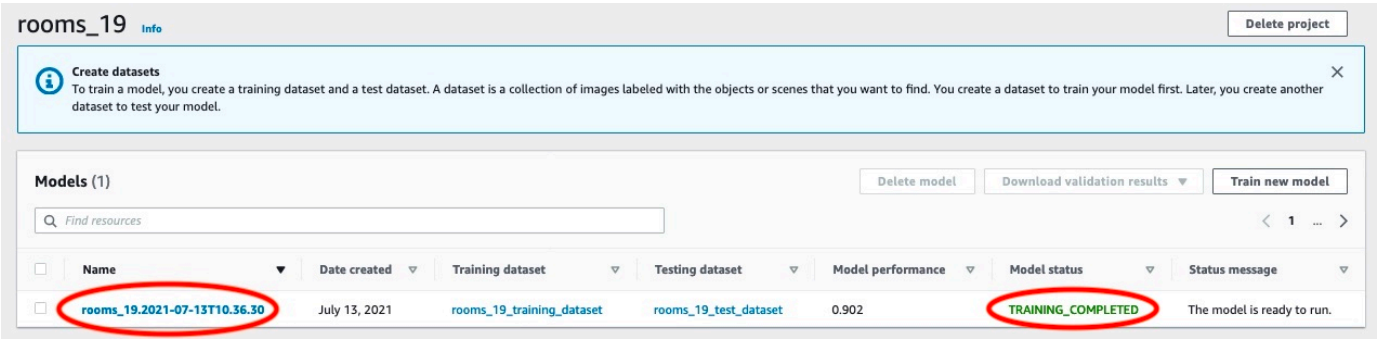
3. En el cuadro de diálogo ¿Quiere entrenar su modelo?, escoja Entrenar modelo.



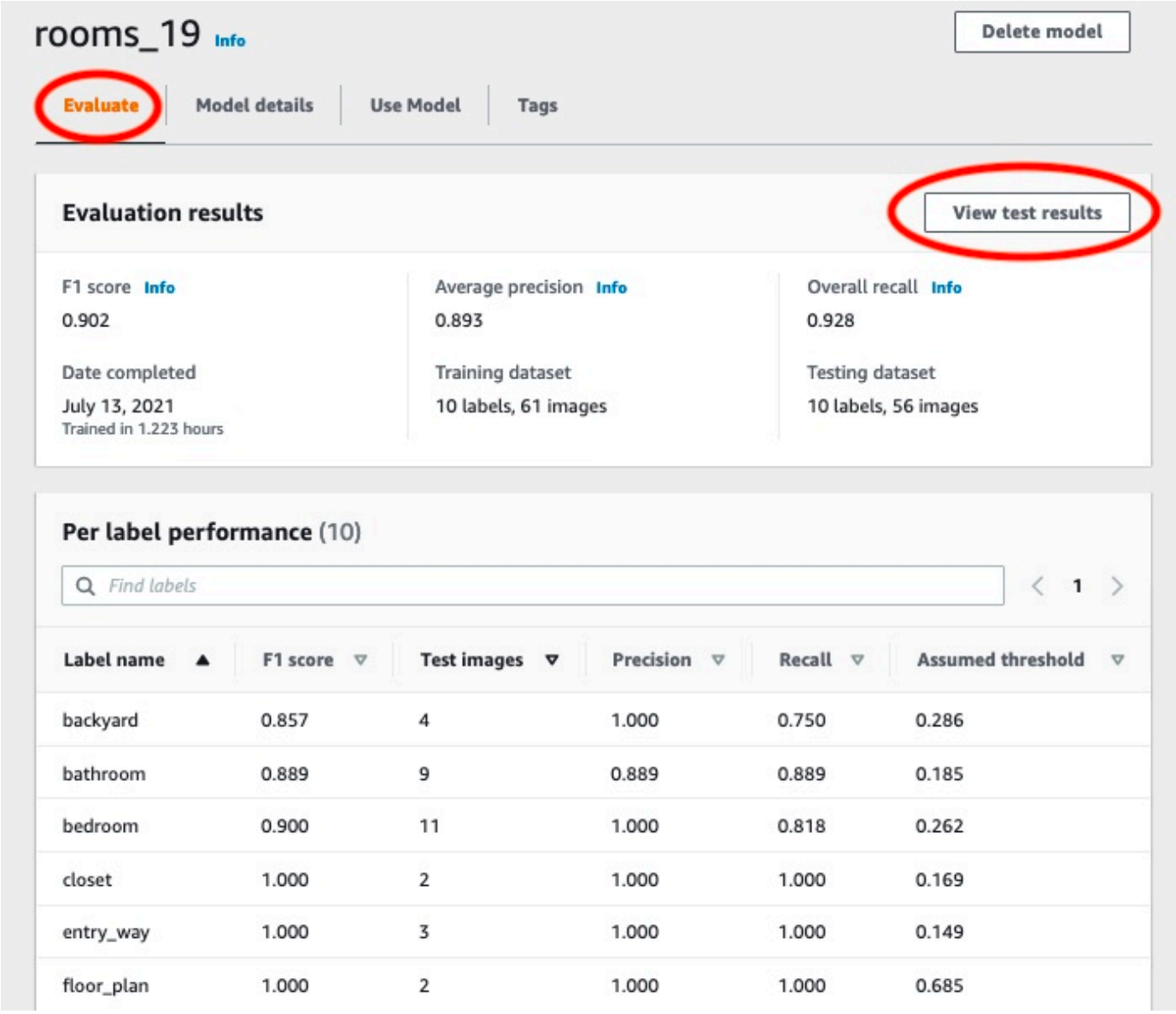
4. En la sección Modelos de la página del proyecto, podrá ver que el entrenamiento en curso. Para comprobar el estado actual, consulte la columna `Model Status` correspondiente a la versión del modelo. El entrenamiento de un modelo tarda un tiempo en completarse.



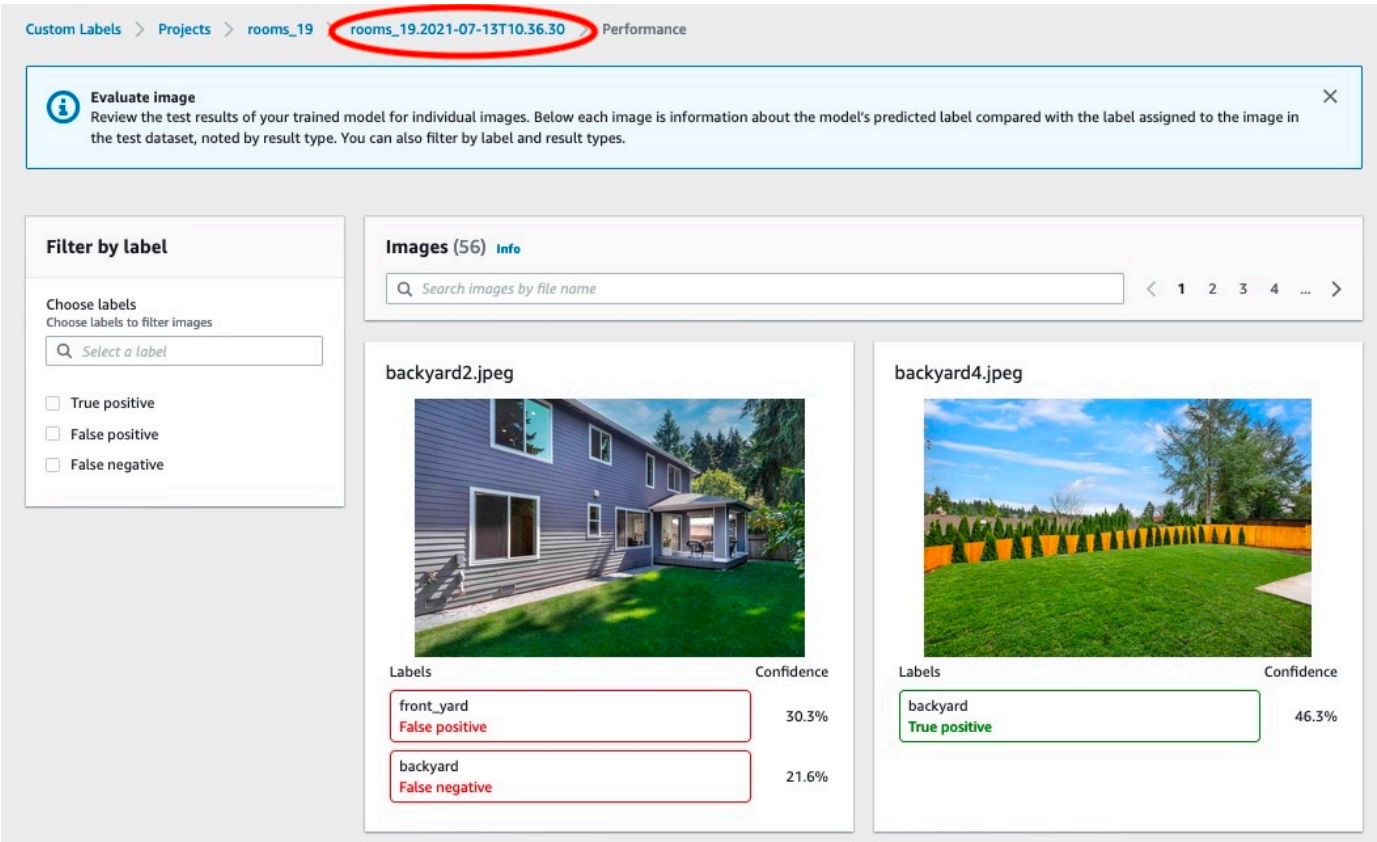
5. Una vez finalizado, elija el nombre del modelo. El entrenamiento termina cuando el estado del modelo es TRAINING_COMPLETED.



6. Pulse el botón Evaluar para ver los resultados de la evaluación. Para obtener información sobre la evaluación de un modelo, consulte Mejora de un modelo de Etiquetas personalizadas de Amazon Rekognition.
7. Seleccione Ver resultados de pruebas para ver los resultados de cada una de las imágenes de prueba. Para obtener más información, consulte Métricas para evaluar su modelo.



8. Tras ver los resultados de las pruebas, elija el nombre del modelo para volver a la página del modelo.



Paso 8: Ejecutar el modelo

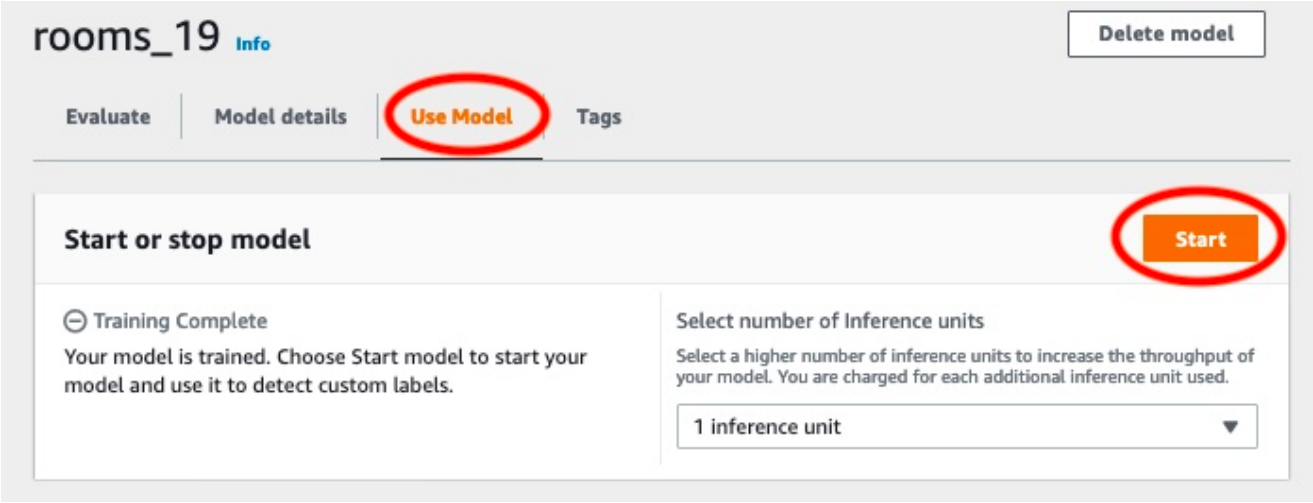
En este paso iniciará el modelo. Después de iniciar el modelo, puede usarlo para analizar nuevas imágenes.

Se le cobrará por la cantidad de tiempo de ejecución del modelo. Detenga el modelo si no necesita analizar imágenes. Puede reiniciar el modelo más adelante. Para obtener más información, consulte Ejecución de un modelo de Etiquetas personalizadas de Amazon Rekognition.

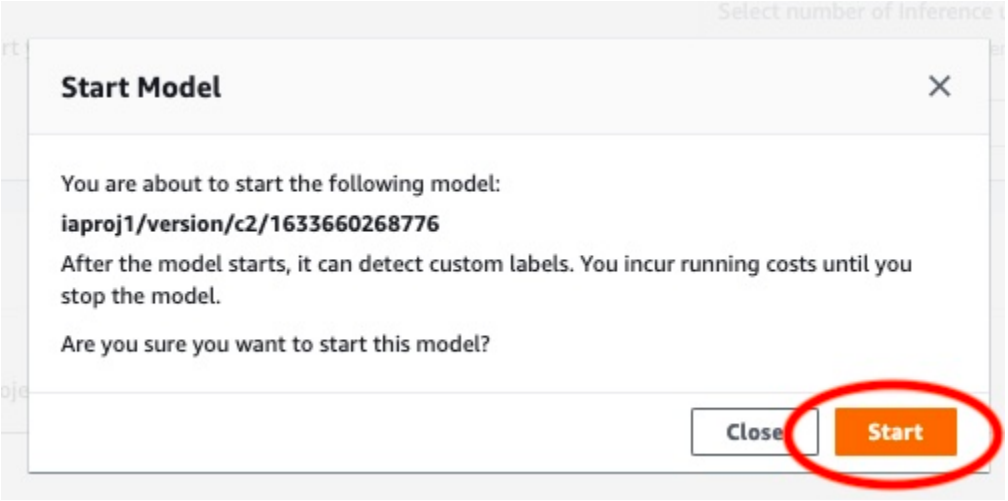
Cómo iniciar el modelo

1. Seleccione la pestaña Usar modelo en la página del modelo.
2. En la sección Iniciar o detener modelo, haga lo siguiente:

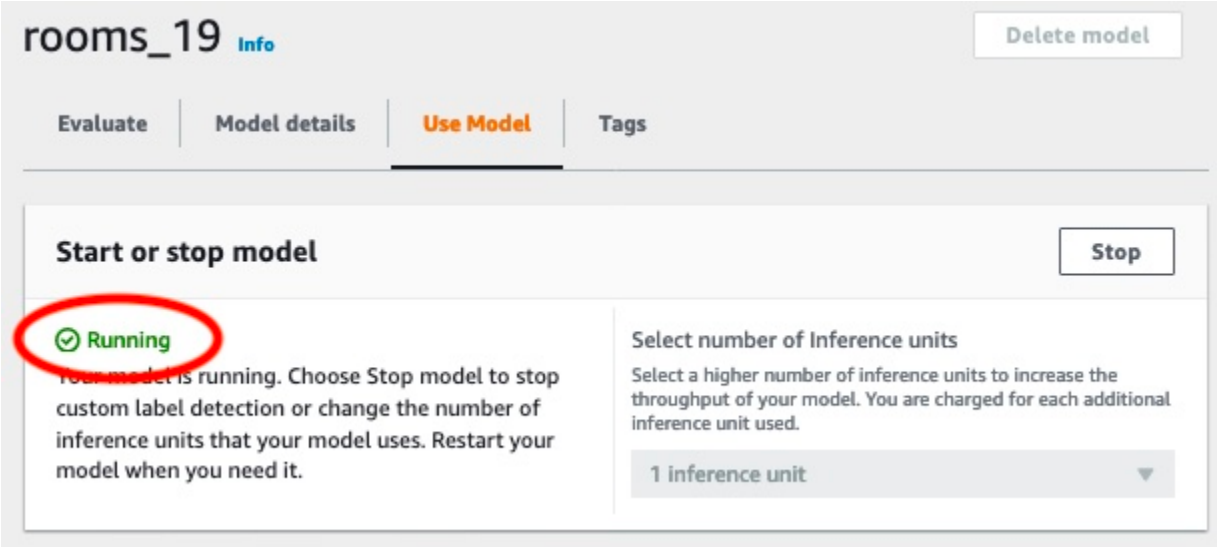
    a. Elija Iniciar.



b. En el cuadro de diálogo Iniciar modelo, seleccione Iniciar.



3. Espere a que se ejecute el modelo. El modelo está funcionando cuando el estado de la sección Iniciar o detener modelo tiene la opción En ejecución activada.



Paso 9: Analizar una imagen con su modelo

Para analizar una imagen, llame a la API. DetectCustomLabels En este paso, utilizas el comando `detect-custom-labels` AWS Command Line Interface (AWS CLI) para analizar una imagen de ejemplo. El AWS CLI comando se obtiene de la consola Amazon Rekognition Custom Labels. La consola configura el AWS CLI comando para usar su modelo. Solo necesita facilitar una imagen que esté almacenada en un bucket de Amazon S3.

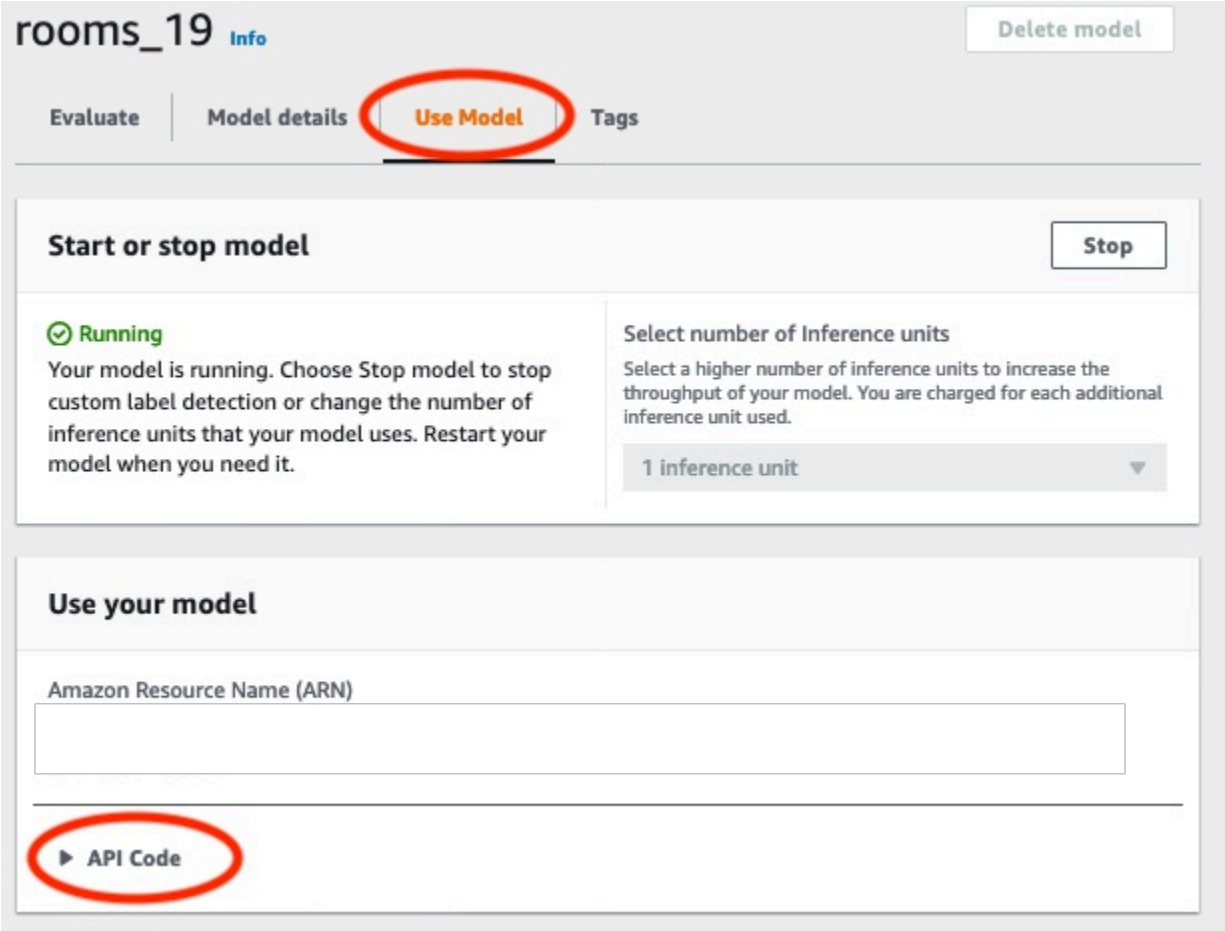
> **Note**
>
> La consola también ofrece un código de ejemplo de Python.

El resultado de `detect-custom-labels` incluye una lista de las etiquetas que se encuentran en la imagen, cuadros delimitadores (si el modelo encuentra ubicaciones de objetos) y la confianza que el modelo tiene en la precisión de las predicciones.

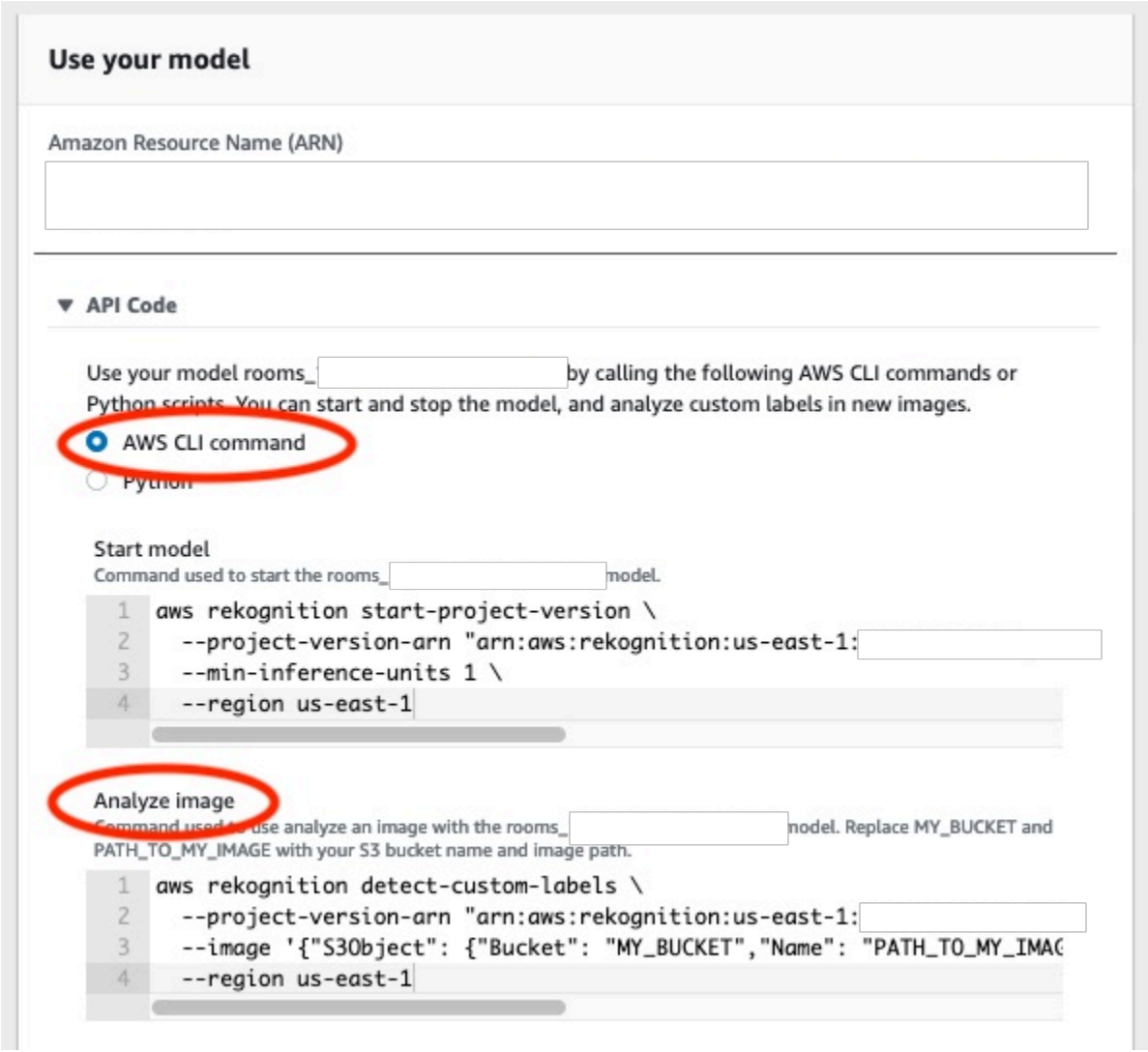
Para obtener más información, consulte Análisis de una imagen con un modelo entrenado.

Cómo analizar una imagen (consola)

1. Si aún no lo ha hecho, configure. AWS CLI Para obtener instrucciones, consulte the section called “Paso 4: Configure y AWS CLIAWS SDKs”.
2. Seleccione la pestaña Usar modelo y luego elija Código de API.



3. Elija Comando AWS CLI.
4. En la sección Analizar imagen, copia el AWS CLI comando que llama`detect-custom-labels`.



5. Cargue una imagen en un bucket de Amazon S3. Para ver las instrucciones, consulte Carga de objetos en Amazon S3 en la Guía del usuario de Amazon Simple Storage Service. Si va a usar imágenes del proyecto Habitaciones, utilice una de las imágenes que haya movido a una carpeta independiente en Paso 1: Reunir las imágenes.

6. En la línea de comandos, introduzca el AWS CLI comando que copió en el paso anterior. Debería ser similar al ejemplo siguiente.

   El valor de `--project-version-arn` debe ser el nombre de recurso de Amazon (ARN) del modelo. El valor de `--region` debe ser la región de AWS en la que se creó el modelo.

   Cambie `MY_BUCKET` y `PATH_TO_MY_IMAGE` por el bucket de Amazon S3 y la imagen que utilizó en el paso anterior.

Si va a usar el perfil custom-labels-access para obtener las credenciales, añada el parámetro `--profile custom-labels-access`.

```
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
```

El resultado JSON del comando AWS CLI debería tener un aspecto similar al siguiente. `Name` es el nombre de la etiqueta de imagen que detectó el modelo. `Confidence` (0-100) es la confianza del modelo en la precisión de la predicción.

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

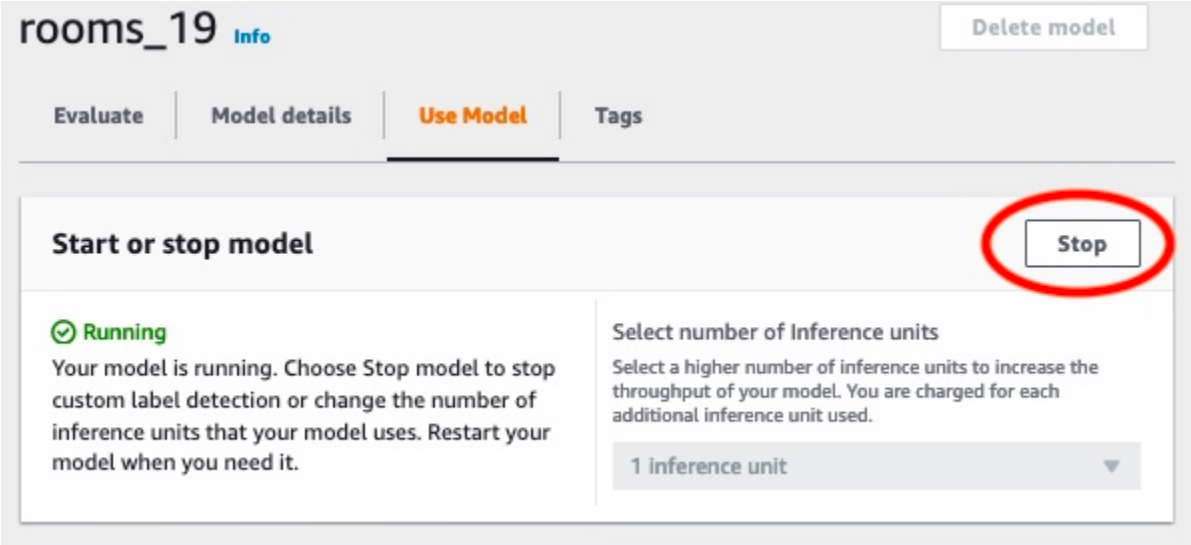
7. Siga utilizando el modelo para analizar otras imágenes. Detenga el modelo si deja de usarlo.

Paso 10: Detener el modelo

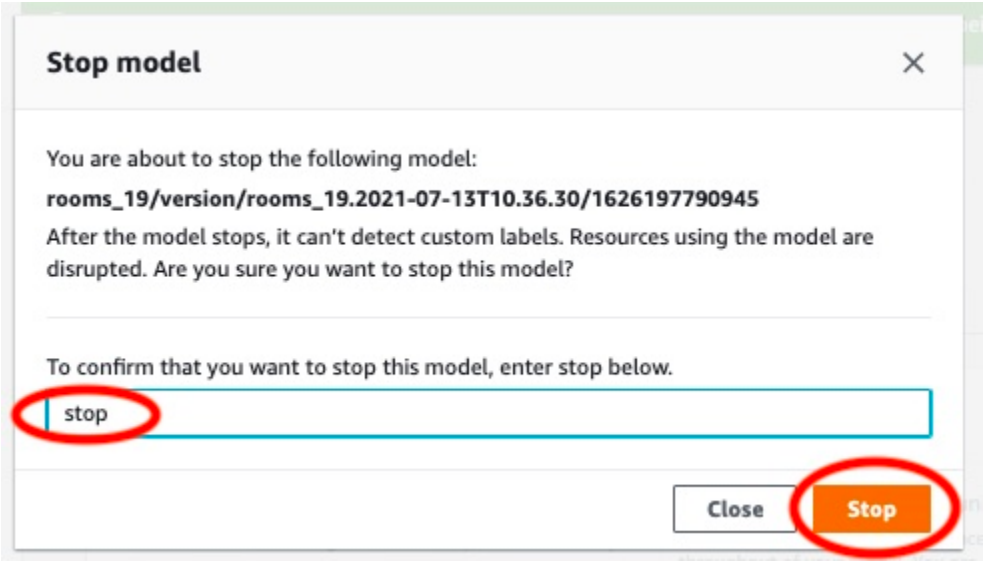
En este paso, dejará de ejecutar el modelo. Se le cobrará por la cantidad de tiempo de ejecución del modelo. Si ha terminado de usar el modelo, debe detenerlo.

Cómo detener su modelo

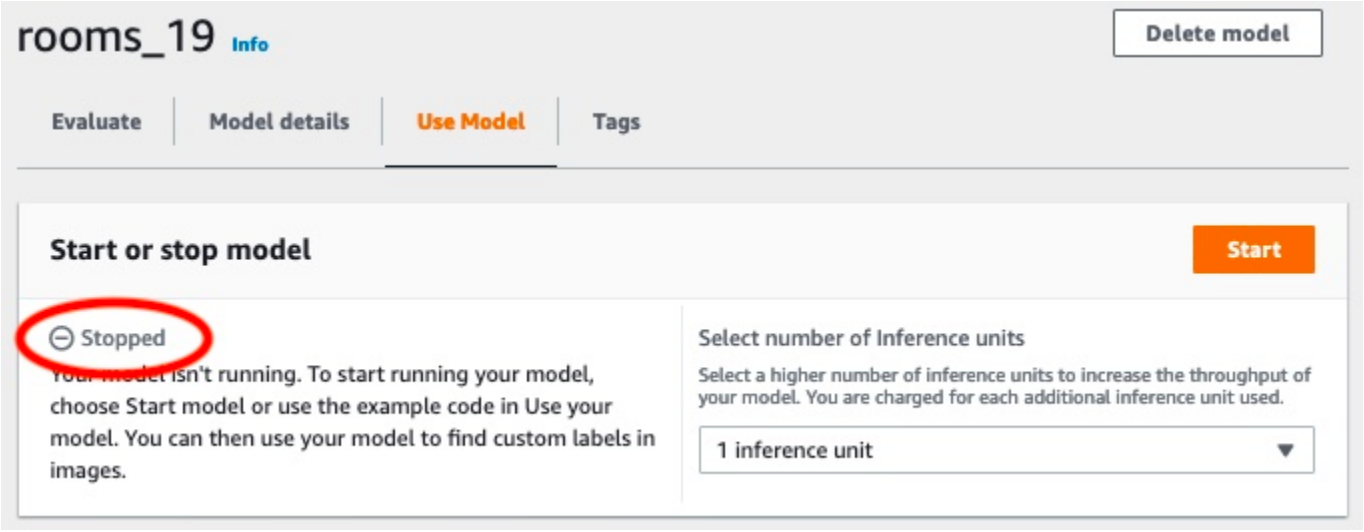
1. En la sección Iniciar o detener modelo, seleccione Detener.



2. En el cuadro de diálogo Detener modelo, escriba detener para confirmar que desea detener el modelo.



3. Seleccione Detener para detener el modelo. El modelo se habrá detenido cuando el estado de la sección Iniciar o detener modelo tenga la opción Detenido activada.


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

Creación de un modelo de Etiquetas personalizadas de Amazon Rekognition

Un modelo es un software que se entrena para encontrar los conceptos, las escenas y los objetos que son exclusivos de su empresa o negocio. Puede crear un modelo con la consola Amazon Rekognition Custom Labels o con el SDK. AWS Antes de crear un modelo de Etiquetas personalizadas de Amazon Rekognition, le recomendamos que lea Qué es Etiquetas personalizadas de Amazon Rekognition.

En esta sección se da información sobre la consola y el SDK para la creación de un proyecto, la creación de conjuntos de datos de entrenamiento y de prueba de distintos tipos de modelos y el entrenamiento de un modelo. En las secciones siguientes, se explica cómo mejorar y utilizar el modelo. Para ver un tutorial sobre cómo crear y utilizar un tipo específico de modelo con la consola, consulte Clasificación de imágenes.

Temas

Creación de un proyecto

Un proyecto permite gestionar las versiones de un modelo y el conjunto de datos de entrenamiento y de prueba de un modelo. Puede crear un modelo a través de la consola de Etiquetas personalizadas de Amazon Rekognition o con la API. Para otras tareas del proyecto, como eliminar un proyecto, consulte Administración de un proyecto de Etiquetas personalizadas de Amazon Rekognition.

Puede utilizar etiquetas para clasificar y administrar sus recursos de Etiquetas personalizadas de Amazon Rekognition, incluidos sus proyectos.

La CreateProjectoperación le permite especificar etiquetas de forma opcional al crear un nuevo proyecto, proporcionando las etiquetas como pares clave-valor que puede usar para categorizar y administrar sus recursos.

Creación de un proyecto de Etiquetas personalizadas de Amazon Rekognition (consola)

Puede usar la consola de Etiquetas personalizadas de Amazon Rekognition para crear un proyecto. La primera vez que utilice la consola en una AWS región nueva, Amazon Rekognition Custom Labels le pedirá que cree un bucket de Amazon S3 (bucket de consola) en su cuenta. AWS Este bucket sirve para almacenar los archivos del proyecto. No puede usar la consola de Etiquetas personalizadas de Amazon Rekognition a menos que se cree el bucket de consola.

Puede usar la consola de Etiquetas personalizadas de Amazon Rekognition para crear un proyecto.

Cómo crear un proyecto (consola)

1. Inicie sesión en la consola Amazon Rekognition AWS Management Console y ábrala en. https://console.aws.amazon.com/rekognition/
2. En el panel izquierdo, elija Usar etiquetas personalizadas. Se abrirá la página de inicio de Etiquetas personalizadas de Amazon Rekognition.
3. En la página de inicio de Etiquetas personalizadas de Amazon Rekognition, seleccione Comenzar.
4. En el panel de navegación izquierdo, elija Proyectos.
5. Elija Crear proyecto.
6. En Nombre del proyecto, asigne un nombre al proyecto.
7. Seleccione Crear proyecto para crearlo.
8. Siga los pasos que se indican en Creación de conjuntos de datos de entrenamiento y prueba a continuación para crear los conjuntos de datos de entrenamiento y de prueba para su proyecto.

Creación de un proyecto de Etiquetas personalizadas de Amazon Rekognition (SDK)

Puede crear un proyecto de Amazon Rekognition Custom Labels llamando. CreateProject La respuesta será un nombre de recurso de Amazon (ARN) que identifica el proyecto. Después de crear un proyecto, se crean los conjuntos de datos para entrenar y probar un modelo. Para obtener más información, consulte Creación de conjuntos de datos de entrenamiento y prueba.

Para crear un proyecto (SDK)

1. Si aún no lo ha hecho, instale y configure el y el AWS CLI . AWS SDKs Para obtener más información, consulte Paso 4: Configure y AWS CLIAWS SDKs.
2. Utilice el siguiente código para crear un proyecto.

   AWS CLI

   En el siguiente ejemplo se ve cómo se crea un proyecto y aparece el ARN.

   Cambie el valor de `project-name` por el nombre del proyecto que desee crear.

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

   Python

   En el siguiente ejemplo se ve cómo se crea un proyecto y aparece el ARN. Indique los siguientes argumentos de línea de comandos:

   - `project_name`: el nombre del proyecto que desea crear.

```
# 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

En el siguiente ejemplo se ve cómo se crea un proyecto y aparece el ARN.

Indique el siguiente argumento de línea de comandos:

- `project_name`: el nombre del proyecto que desea crear.

```
/*
   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. Recuerde el nombre del ARN del proyecto que aparece en la respuesta. Lo necesitará para crear un modelo.
4. Siga los pasos que se indican en Crear conjuntos de datos de entrenamiento y prueba (SDK) a continuación para crear los conjuntos de datos de entrenamiento y de prueba para su proyecto.

CreateProject solicitud de operación

El formato de la solicitud de CreateProject operación es el siguiente:

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

```
}
```

Creación de conjuntos de datos de entrenamiento y prueba

Un conjunto de datos es un conjunto de imágenes y etiquetas que describen esas imágenes. El proyecto necesita un conjunto de datos de entrenamiento y un conjunto de datos de prueba. Etiquetas personalizadas de Amazon Rekognition utiliza el conjunto de datos de entrenamiento para entrenar un modelo. Tras el entrenamiento, Etiquetas personalizadas de Amazon Rekognition utiliza el conjunto de datos de prueba para comprobar en qué medida el modelo entrenado predice las etiquetas correctas.

Puede crear conjuntos de datos con la consola Amazon Rekognition Custom Labels o con el SDK. AWS Antes de crear un conjunto de datos, le recomendamos que lea Qué es Etiquetas personalizadas de Amazon Rekognition. En el caso de otras tareas relacionadas con los conjuntos de datos, consulte Administración de conjuntos de datos.

Los pasos para crear conjuntos de datos de entrenamiento y prueba para un proyecto son los siguientes:

Cómo crear conjuntos de datos de entrenamiento y prueba para su proyecto

1. Determine cómo debe etiquetar sus conjuntos de datos de entrenamiento y prueba. Para obtener más información, Finalidad de los conjuntos de datos.
2. Recopile las imágenes para sus conjuntos de datos de entrenamiento y prueba. Para obtener más información, consulte the section called “Preparación de imágenes”.
3. Cree los conjuntos de datos de entrenamiento y de prueba. Para obtener más información, consulte Creación de conjuntos de datos de entrenamiento y prueba. Si utiliza el SDK, consulte. AWS Crear conjuntos de datos de entrenamiento y prueba (SDK)
4. Si es necesario, agregue etiquetas de tipo imagen o cuadros delimitadores a las imágenes de su conjunto de datos. Para obtener más información, consulte Etiquetado de imágenes.

Después de crear los conjuntos de datos, puede entrenar el modelo.

Temas

- Finalidad de los conjuntos de datos

- Preparación de imágenes
- Creación de conjuntos de datos de entrenamiento y prueba
- Etiquetado de imágenes
- Depuración de errores de conjuntos de datos

Finalidad de los conjuntos de datos

La forma en que etiquete los conjuntos de datos de entrenamiento y prueba de su proyecto determina el tipo de modelo que se va a crear. Con Etiquetas personalizadas de Amazon Rekognition, puede crear modelos que hagan lo siguiente.

- Encontrar objetos, escenas y conceptos
- Encontrar ubicaciones de objetos
- Encontrar ubicaciones de marcas

Encontrar objetos, escenas y conceptos

El modelo clasifica los objetos, las escenas y los conceptos que están asociados a una imagen completa.

Puede crear dos tipos de modelo de clasificación: clasificación de imágenes y clasificación de etiquetas múltiples. Para ambos tipos de modelo de clasificación, el modelo busca una o varias etiquetas que coincidan del conjunto entero de etiquetas utilizado para realizar el entrenamiento. Tanto los conjuntos de datos de entrenamiento como los de prueba necesitan al menos dos etiquetas.

Clasificación de imágenes

El modelo clasifica las imágenes como pertenecientes a un conjunto de etiquetas predefinidas. Por ejemplo, si desea un modelo que determine si en la imagen hay una sala de estar. La siguiente imagen puede tener una etiqueta de imagen sala_estar.

Para este tipo de modelo, agregue una sola etiqueta de imagen a cada una de las imágenes del conjunto de datos de entrenamiento y prueba. Para ver un objeto de ejemplo, consulte Clasificación de imágenes.

Clasificación de etiquetas múltiples

El modelo clasifica las imágenes en varias categorías, como por ejemplo, el tipo de flor y si tiene hojas o no. Por ejemplo, la imagen siguiente puede tener las etiquetas de imagen euforbio_mediterráneo y sin_hojas.

Para este tipo de modelo, asigne etiquetas de imagen por cada categoría a las imágenes del conjunto de datos de entrenamiento y prueba. Para ver un objeto de ejemplo, consulte Clasificación de imágenes de etiquetas múltiples.

Asignación de etiquetas de imagen

Si las imágenes están almacenadas en un bucket de Amazon S3, puede utilizar los nombres de las carpetas para agregar automáticamente etiquetas de imagen. Para obtener más información, consulte Importar imágenes desde un bucket de Amazon S3. También puede agregar etiquetas de imagen a las imágenes después de crear un conjunto de datos. Para obtener más información, consulte the section called “Asignación de etiquetas de imagen a una imagen”. Puede agregar etiquetas nuevas cada vez que las necesite. Para obtener más información, consulte Administración de etiquetas.

Encontrar ubicaciones de objetos

Para crear un modelo que prediga la ubicación de los objetos en las imágenes, defina los cuadros delimitadores de la ubicación de los objetos y las etiquetas para las imágenes en los conjuntos de datos de entrenamiento y prueba. Un cuadro delimitador es un cuadro que rodea ajustadamente un objeto. Por ejemplo, en la siguiente imagen se ven unos cuadros delimitadores alrededor de un Amazon Echo y un Amazon Echo Dot. Cada cuadro delimitador tiene una etiqueta asignada (Amazon Echo o Amazon Echo Dot).

Para encontrar las ubicaciones de los objetos, los conjuntos de datos necesitan al menos una etiqueta. Mientras se entrena el modelo, se crea automáticamente otra etiqueta que representa el área situada fuera de los cuadros delimitadores de una imagen.

Asignación de cuadros delimitadores

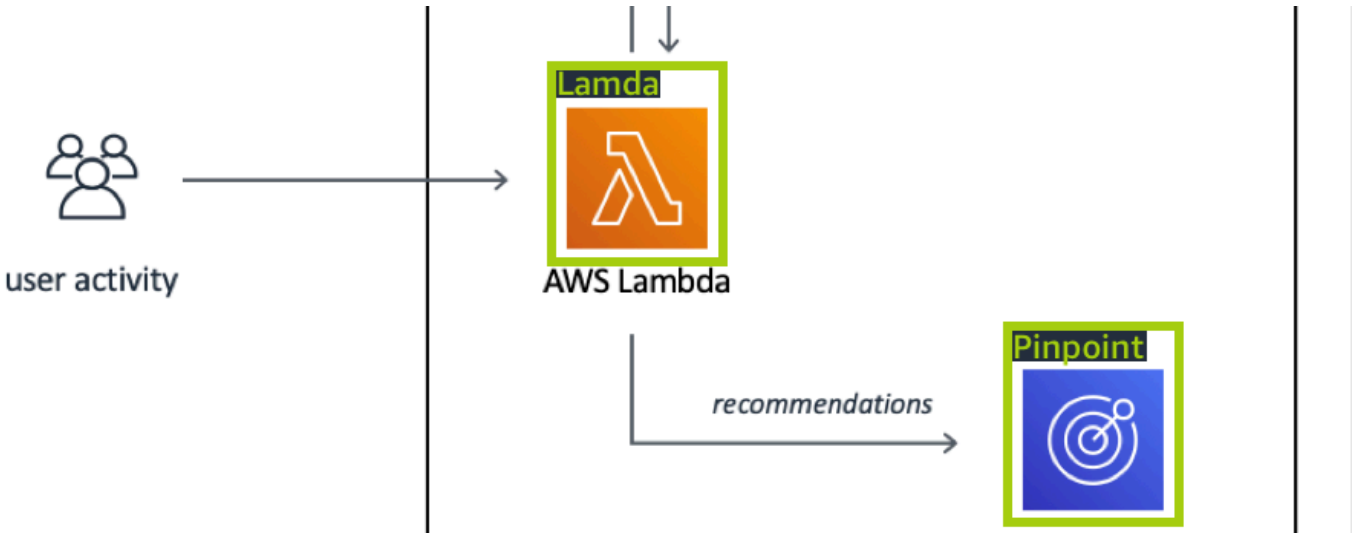
Al crear el conjunto de datos, puede incluir información sobre los cuadros delimitadores de las imágenes. Por ejemplo, puedes importar un archivo de manifiesto en formato SageMaker AI Ground Truth que contenga cuadros delimitadores. También puede agregar cuadros delimitadores después de crear un conjunto de datos. Para obtener más información, consulte Etiquetado de objetos con cuadros delimitadores. Puede agregar etiquetas nuevas cada vez que las necesite. Para obtener más información, consulte Administración de etiquetas.

Encontrar ubicaciones de marcas

Si quiere encontrar la ubicación de marcas, como logotipos y personajes animados, puede usar dos tipos diferentes de imágenes para las imágenes del conjunto de datos de entrenamiento.

- Imágenes que son solo de logotipo. Cada imagen necesita una única etiqueta de imagen que represente el nombre del logotipo. Por ejemplo, la etiqueta de imagen de la siguiente imagen podría ser Lambda.

- Imágenes con el logotipo en ubicaciones naturales, como un partido de fútbol o un diagrama arquitectónico. Cada imagen de entrenamiento necesita cuadros delimitadores que rodeen cada instancia del logotipo. Por ejemplo, en la siguiente imagen se muestra un diagrama de arquitectura con recuadros delimitadores etiquetados que rodean los logotipos de AWS Lambda y Amazon Pinpoint.



Le recomendamos que no mezcle etiquetas de imagen y cuadros delimitadores en las imágenes de entrenamiento.

Las imágenes de prueba deben tener cuadros delimitadores alrededor de las imágenes de la marca que quiera encontrar. Puede dividir el conjunto de datos de entrenamiento para crear el conjunto de datos de prueba, solo si las imágenes de entrenamiento incluyen cuadros delimitadores etiquetados. Si las imágenes de entrenamiento solo tienen etiquetas de imagen, debe crear un conjunto de conjuntos de datos de prueba que incluya imágenes con cuadros delimitadores etiquetados. Si se entrena un modelo para que busque ubicaciones de marcas, haga Etiquetado de objetos con cuadros delimitadores y Asignación de etiquetas de imagen a una imagen según la forma en que etiquete las imágenes.

En el proyecto de ejemplo Detección de marcas, se ve cómo Etiquetas personalizadas de Amazon Rekognition utiliza cuadros delimitadores etiquetados para entrenar un modelo que busca ubicaciones de objetos.

Requisitos de etiquetado de tipos de modelos

Use la siguiente tabla para determinar cómo etiquetar las imágenes.

Puede combinar etiquetas de imagen e imágenes etiquetadas con cuadros delimitadores en un único conjunto de datos. En este caso, Etiquetas personalizadas de Amazon Rekognition elige si desea crear un modelo de imagen o un modelo de ubicación de objetos.

Ejemplo	Imágenes de entrenamiento	Imágenes de prueba
Clasificación de imágenes	1 etiqueta de imagen por imagen	1 etiqueta de imagen por imagen
Clasificación de etiquetas múltiples	Etiquetas múltiples de imagen por imagen	Etiquetas múltiples de imagen por imagen
Encontrar ubicaciones de marcas	Etiquetas de imagen (también puede utilizar cuadros delimitadores etiquetados)	Cuadros delimitadores etiquetados
Encontrar ubicaciones de objetos	Cuadros delimitadores etiquetados	Cuadros delimitadores etiquetados

Preparación de imágenes

Las imágenes del conjunto de datos de entrenamiento y prueba contienen los objetos, escenas o conceptos que desea que encuentre el modelo.

El contenido de las imágenes debe estar formado por diversos fondos e iluminación que representen las imágenes que desea que identifique el modelo entrenado.

En esta sección, se da información acerca de las imágenes del conjunto de datos de entrenamiento y prueba.

Formato de imagen

Puede entrenar los modelos de Etiquetas personalizadas de Amazon Rekognition con imágenes en formato PNG y JPEG. Del mismo modo, para detectar el uso de etiquetas personalizadas con `DetectCustomLabels`, necesitará imágenes en formato PNG y JPEG.

Recomendaciones de imágenes de entrada

Etiquetas personalizadas de Amazon Rekognition necesita imágenes para entrenar y probar el modelo. Para preparar las imágenes, tenga presente lo siguiente:

- Elija un dominio específico para el modelo que desee crear. Por ejemplo, puede elegir un modelo con vistas panorámicas y otro modelo con objetos, como piezas de máquinas. Etiquetas personalizadas de Amazon Rekognition funciona mejor si las imágenes se encuentran en el dominio elegido.
- Utilice al menos 10 imágenes para entrenar el modelo.
- Las imágenes deben estar en formato PNG o JPEG.
- Utilice imágenes que muestren el objeto en diferentes iluminaciones, fondos y resoluciones.
- Las imágenes de entrenamiento y de prueba deben ser similares a las imágenes con las que desea utilizar el modelo.
- Decida qué etiquetas desea asignar a las imágenes.
- Asegúrese de que las imágenes sean lo suficientemente grandes en cuanto a resolución. Para obtener más información, consulte Directrices y cuotas en Etiquetas personalizadas de Amazon Rekognition.
- Asegúrese de que las oclusiones no oculten los objetos que desee detectar.
- Utilice imágenes que tengan un contraste suficiente respecto al fondo.

- Utilice imágenes que sean brillantes y nítidas. Evite en la medida de lo posible utilizar imágenes que puedan resultar borrosas debido al movimiento del sujeto y de la cámara.
- Utilice una imagen en la que el objeto ocupe una gran proporción de la imagen.
- Las imágenes del conjunto de datos de prueba no deberían ser imágenes del conjunto de datos de entrenamiento. Deben incluir los objetos, las escenas y los conceptos para los que el modelo está entrenado para analizar.

Tamaño del conjunto de imágenes

Etiquetas personalizadas de Amazon Rekognition utiliza un conjunto de imágenes para entrenar un modelo. Como mínimo, debe utilizar 10 imágenes para el entrenamiento. Etiquetas personalizadas de Amazon Rekognition necesita imágenes para entrenar y probar el modelo. Para obtener más información, consulte Creación de conjuntos de datos de entrenamiento y prueba.

Creación de conjuntos de datos de entrenamiento y prueba

Puede empezar con un proyecto que tenga un único conjunto de datos o un proyecto que tenga conjuntos de datos de entrenamiento y de prueba independientes. Si empieza con un único conjunto de datos, Etiquetas personalizadas de Amazon Rekognition lo divide durante el entrenamiento para crear un conjunto de datos de entrenamiento (80 %) y un conjunto de datos de prueba (20 %) para su proyecto. Comience con un único conjunto de datos si quiere que Etiquetas personalizadas de Amazon Rekognition decida dónde se utilizan las imágenes para el entrenamiento y las pruebas. Para tener un control total sobre el entrenamiento, las pruebas y el nivel de rendimiento, le recomendamos que inicie el proyecto con conjuntos de datos de entrenamiento y prueba independientes.

Puede crear conjuntos de datos de entrenamiento y prueba para un proyecto importando imágenes a través de una de las siguientes ubicaciones:

- Importar imágenes desde un bucket de Amazon S3
- Importar imágenes desde un ordenador local
- Utilizar un archivo de manifiesto para importar imágenes
- Copia de contenido de un conjunto de datos existente

Si comienza su proyecto con conjuntos de datos de entrenamiento y prueba separados, podrá usar diferentes ubicaciones de origen para cada conjunto de datos.

Según el lugar desde el que importe las imágenes, es posible que no estén etiquetadas. Por ejemplo, las imágenes importadas de un ordenador local no están etiquetadas. Las imágenes importadas de un archivo de manifiesto de Amazon SageMaker AI Ground Truth están etiquetadas. Puede utilizar la consola de Etiquetas personalizadas de Amazon Rekognition para agregar, cambiar y asignar etiquetas. Para obtener más información, consulte Etiquetado de imágenes.

Si las imágenes se cargan con errores, faltan imágenes o faltan etiquetas en las imágenes, lea Depuración de un modelo de entrenamiento con errores.

Para obtener más información sobre los conjuntos de datos, consulte Administración de conjuntos de datos.

Crear conjuntos de datos de entrenamiento y prueba (SDK)

Puede usar el AWS SDK para crear conjuntos de datos de entrenamiento y prueba.

La operación `CreateDataset` le permite especificar etiquetas de forma opcional al crear un nuevo conjunto de datos con el fin de categorizar y administrar sus recursos.

Conjunto de datos de entrenamiento

Puedes usar el AWS SDK para crear un conjunto de datos de entrenamiento de las siguientes maneras.

- CreateDatasetUtilícelo con un archivo de manifiesto en formato Amazon Sagemaker que proporcione. Para obtener más información, consulte the section called “Creación de un archivo de manifiesto”. Para ver el código de ejemplo, consulte Creación de un conjunto de datos con un archivo de manifiesto (SDK) de SageMaker AI Ground Truth.
- Use `CreateDataset` para copiar un conjunto de datos existente de Etiquetas personalizadas de Amazon Rekognition. Para ver el código de ejemplo, consulte Creación de un conjunto de datos mediante un conjunto de datos existente (SDK).
- Cree un conjunto de datos vacío con `CreateDataset` y agregue entradas del conjunto de datos más adelante con UpdateDatasetEntries. Para crear un conjunto de datos vacío, consulte Agregar un conjunto de datos a un proyecto. Para agregar imágenes a un conjunto de datos, consulte Agregar más imágenes (SDK). Debe agregar las entradas del conjunto de datos antes de poder entrenar un modelo.

Conjunto de datos de prueba

Puede usar el AWS SDK para crear un conjunto de datos de prueba de las siguientes maneras:

- CreateDatasetUtilícelo con un archivo de manifiesto en formato Amazon Sagemaker que proporcione. Para obtener más información, consulte the section called “Creación de un archivo de manifiesto”. Para ver el código de ejemplo, consulte Creación de un conjunto de datos con un archivo de manifiesto (SDK) de SageMaker AI Ground Truth.
- Use `CreateDataset` para copiar un conjunto de datos existente de Etiquetas personalizadas de Amazon Rekognition. Para ver el código de ejemplo, consulte Creación de un conjunto de datos mediante un conjunto de datos existente (SDK).
- Cree un conjunto de datos vacío con `CreateDataset` y agregue entradas del conjunto de datos más adelante con `UpdateDatasetEntries`. Para crear un conjunto de datos vacío, consulte Agregar un conjunto de datos a un proyecto. Para agregar imágenes a un conjunto de datos, consulte Agregar más imágenes (SDK). Debe agregar las entradas del conjunto de datos antes de poder entrenar un modelo.
- Divida el conjunto de datos de entrenamiento en conjuntos de datos de entrenamiento y prueba independientes. En primer lugar, cree un conjunto de datos de prueba vacío con `CreateDataset`. A continuación, mueva el 20% de las entradas del conjunto de datos de entrenamiento al conjunto de datos de prueba mediante una llamada DistributeDatasetEntries. Para crear un conjunto de datos vacío, consulte Agregar un conjunto de datos a un proyecto (SDK). Para dividir el conjunto de datos de entrenamiento, consulte Distribución de un conjunto de datos de entrenamiento (SDK).

Importar imágenes desde un bucket de Amazon S3

Las imágenes se importan desde un bucket de Amazon S3. Puede usar el bucket de consola u otro bucket de Amazon S3 de su AWS cuenta. Si utiliza el bucket de consola, los permisos necesarios ya estarán configurados. Si no utiliza el bucket de consola, consulte Acceso a buckets de Amazon S3 externos.

> **Note**
>
> No puede usar el AWS SDK para crear un conjunto de datos directamente a partir de las imágenes de un bucket de Amazon S3. En su lugar, cree un archivo de manifiesto que haga referencia a las ubicaciones de origen de las imágenes. Para obtener más información, consulte Utilizar un archivo de manifiesto para importar imágenes

Durante la creación del conjunto de datos, puede optar por asignar nombres de etiquetas a las imágenes en función del nombre de la carpeta que contiene las imágenes. Las carpetas deben

ser secundarias en la ruta de carpeta de Amazon S3 que indique en la ubicación de la carpeta S3 durante la creación del conjunto de datos. Para crear un conjunto de datos, consulte Crear un conjunto de datos mediante la importación de imágenes de un bucket de S3.

Por ejemplo, supongamos que un bucket de Amazon S3 tiene la siguiente estructura de carpetas. Si indica la ubicación de la carpeta de Amazon S3 como S3-bucket/alexa-devices, a las imágenes de la carpeta echo se les asigna la etiqueta echo. Del mismo modo, a las imágenes de la carpeta echo-dot se les asigna la etiqueta echo-dot. Los nombres de las carpetas secundarias de más abajo no se utilizan para etiquetar las imágenes. En su lugar, se utiliza la carpeta secundaria correspondiente a la ubicación de la carpeta de Amazon S3. Por ejemplo, a las imágenes de la carpeta white-echo-dotsse les asigna la etiqueta echo-dot. Las imágenes que se encuentran en el nivel de la ubicación de la carpeta S3 (alexa-devices) no tienen etiquetas asignadas.

Las carpetas de más abajo de la estructura de carpetas se pueden usar para etiquetar las imágenes si se indica una ubicación más abajo de la carpeta S3. Por ejemplo, si especifica S3- bucket/alexa-devices/echo -dot, las imágenes de la carpeta white-echo-dotse etiquetan. white-echo-dot Las imágenes que se encuentren fuera de la ubicación de la carpeta s3 indicada, como echo, no se importan.

```
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
        ### .
        ### .
```

Le recomendamos que utilice el bucket de Amazon S3 (bucket de consola) que Amazon Rekognition creó para usted la primera vez que abrió la consola en la región actual. AWS Si el bucket de Amazon S3 que está utilizando es diferente (externo) al bucket de la consola, la consola le solicitará que

configure los permisos adecuados durante la creación del conjunto de datos. Para obtener más información, consulte the section called “Paso 2: Configurar los permisos de la consola”.

Crear un conjunto de datos mediante la importación de imágenes de un bucket de S3

En el siguiente procedimiento, se explica cómo crear un conjunto de datos utilizando imágenes almacenadas en el bucket de consola de S3. Las imágenes se etiquetan automáticamente con el nombre de la carpeta en la que se almacenan.

Una vez importadas las imágenes, puede agregar más imágenes, asignar etiquetas y agregar cuadros delimitadores en la página de la galería de un conjunto de datos. Para obtener más información, consulte Etiquetado de imágenes.

Cargar las imágenes en un bucket de Amazon Simple Storage Service

1. Cree una carpeta en el sistema de archivos local. Use un nombre de carpeta como alexa-devices.
2. Dentro de la carpeta que acabe de crear, cree carpetas con el nombre de cada etiqueta que quiera usar. Por ejemplo, echo y echo-dot. La estructura debería tener un aspecto similar al siguiente.

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

3. Coloque las imágenes que se correspondan con una etiqueta en la carpeta con el mismo nombre de etiqueta.
4. Inicie sesión en la consola de Amazon S3 AWS Management Console y ábrala en https://console.aws.amazon.com/s3/.
5. Añada la carpeta que creó en el paso 1 al bucket de Amazon S3 (bucket de consola) que Etiquetas personalizadas de Amazon Rekognition crearon para usted durante la primera

configuración. Para obtener más información, consulte Administración de un proyecto de Etiquetas personalizadas de Amazon Rekognition.

6. Abra la consola Amazon Rekognition en. https://console.aws.amazon.com/rekognition/
7. Elija Usar etiquetas personalizadas.
8. Elija Comenzar.
9. En el panel de navegación izquierdo, elija Proyectos.
10. En la página Proyectos, elija el proyecto al que desee añadir el conjunto de datos. Se abrirá la página de detalles del proyecto.
11. Elija Crear conjunto de datos. Se abrirá la página Crear conjunto de datos.
12. En Configuración inicial, seleccione Empezar con un único conjunto de datos o Empezar con un conjunto de datos de entrenamiento. Para crear un modelo de mayor calidad, le recomendamos empezar con conjuntos de datos de entrenamiento y de prueba independientes.

    Single dataset

    a. En la sección Detalles del conjunto de datos de entrenamiento, seleccione Importar imágenes del bucket de S3.
    b. En la sección Detalles del conjunto de datos de entrenamiento, introduzca la información de los pasos 13 al 15 de la sección de Configuración de la fuente de la imagen.

    Separate training and test datasets

    a. En la sección Detalles del conjunto de datos de entrenamiento, seleccione Importar imágenes del bucket de S3.
    b. En la sección Detalles del conjunto de datos de entrenamiento, introduzca la información de los pasos 13 al 15 de la sección de Configuración de la fuente de la imagen.
    c. En la sección Detalles del conjunto de datos de prueba, seleccione Importar imágenes del bucket de S3.
    d. En la sección Detalles del conjunto de datos de prueba, introduzca la información de los pasos 13 al 15 de la sección de Configuración de la fuente de la imagen.
13. Seleccione Importar imágenes del bucket de Amazon S3.
14. En el URI de S3, introduzca la ubicación del bucket de Amazon S3 y la ruta de la carpeta.
15. Seleccione Adjuntar etiquetas automáticamente a las imágenes en función de la carpeta.
16. Elija Crear conjuntos de datos. Se abrirá la página de los conjuntos de datos de su proyecto.

17. Si necesita agregar o cambiar etiquetas, consulte Etiquetado de imágenes.
18. Siga los pasos que se indican en Entrenamiento de un modelo (consola) para entrenar el modelo.

Importar imágenes desde un ordenador local

Las imágenes se cargan directamente desde el ordenador. Puede cargar hasta 30 imágenes a la vez.

Las imágenes que suba no tendrán etiquetas asociadas. Para obtener más información, consulte Etiquetado de imágenes. Si tiene que cargar muchas imágenes, considere la posibilidad de utilizar un bucket de Amazon S3. Para obtener más información, consulte Importar imágenes desde un bucket de Amazon S3.

> **Note**
>
> No puede usar el AWS SDK para crear un conjunto de datos con imágenes locales. En su lugar, cree un archivo de manifiesto y cargue las imágenes en un bucket de Amazon S3. Para obtener más información, consulte Utilizar un archivo de manifiesto para importar imágenes.

Para crear un conjunto de datos con imágenes en un equipo local (consola)

1. Abra la consola Amazon Rekognition en. https://console.aws.amazon.com/rekognition/
2. Elija Usar etiquetas personalizadas.
3. Elija Comenzar.
4. En el panel de navegación izquierdo, elija Proyectos.
5. En la página Proyectos, elija el proyecto al que desee añadir el conjunto de datos. Se abrirá la página de detalles del proyecto.
6. Elija Crear conjunto de datos. Se abrirá la página Crear conjunto de datos.
7. En Configuración inicial, seleccione Empezar con un único conjunto de datos o Empezar con un conjunto de datos de entrenamiento. Para crear un modelo de mayor calidad, le recomendamos empezar con conjuntos de datos de entrenamiento y de prueba independientes.

Single dataset

a. En la sección de detalles del conjunto de datos de entrenamiento, seleccione Cargar imágenes del ordenador.
b. Elija Crear conjunto de datos.
c. En la página del conjunto de datos del proyecto, seleccione Agregar imágenes.
d. Escoja las imágenes que quiera cargar en el conjunto de datos de los archivos del ordenador. Puede arrastrar las imágenes o elegir las imágenes deseadas de su equipo local.
e. Seleccione Cargar imágenes.

Separate training and test datasets

a. En la sección de detalles del conjunto de datos de entrenamiento, seleccione Cargar imágenes del ordenador.
b. En la sección Detalles del conjunto de datos de prueba, seleccione Cargar imágenes desde el equipo.

> **Note**
>
> Los conjuntos de datos de entrenamiento y de prueba pueden tener diferentes fuentes de imágenes.

c. Elija Crear conjuntos de datos. La página de conjuntos de datos del proyecto incluye la pestaña Entrenamiento y la pestaña Prueba para los conjuntos de datos respectivos.
d. Seleccione Acciones y luego Agregar imágenes al conjunto de datos de entrenamiento.
e. Escoja las imágenes que quiera cargar en el conjunto de datos. Puede arrastrar las imágenes o elegir las imágenes deseadas de su equipo local.
f. Seleccione Cargar imágenes.
g. Repita los pasos del 5e al 5g. En el paso 5e, seleccione Acciones y luego Agregar imágenes al conjunto de datos de entrenamiento.

8. Siga los pasos que se indican en Etiquetado de imágenes para etiquetar las imágenes.
9. Siga los pasos que se indican en Entrenamiento de un modelo (consola) para entrenar su modelo.

Utilizar un archivo de manifiesto para importar imágenes

Puede crear un conjunto de datos con un archivo de manifiesto con el formato Amazon SageMaker AI Ground Truth. Puedes usar el archivo de manifiesto de un trabajo de Amazon SageMaker AI Ground Truth. Si tus imágenes y etiquetas no tienen el formato de un archivo de manifiesto de SageMaker AI Ground Truth, puedes crear un archivo de manifiesto en formato SageMaker AI y usarlo para importar las imágenes etiquetadas.

La operación `CreateDataset` se actualiza para que pueda especificar etiquetas de forma opcional al crear un nuevo conjunto de datos. Las etiquetas son pares clave-valor que puede usar para categorizar y administrar sus recursos.

Temas

- Creación de un conjunto de datos con un archivo de manifiesto de SageMaker AI Ground Truth (consola)
- Creación de un conjunto de datos con un archivo de manifiesto (SDK) de SageMaker AI Ground Truth
- Crear una solicitud de conjunto de datos
- Etiquetar imágenes con un trabajo de Amazon SageMaker AI Ground Truth
- Creación de un archivo de manifiesto
- Importación de etiquetas de imagen en los archivos de manifiesto
- Localización de objetos en archivos de manifiesto
- Reglas de validación de archivos de manifiesto
- Convertir otros formatos de conjunto de datos en un archivo de manifiesto

Creación de un conjunto de datos con un archivo de manifiesto de SageMaker AI Ground Truth (consola)

El siguiente procedimiento muestra cómo crear un conjunto de datos mediante un archivo de manifiesto en formato SageMaker AI Ground Truth.

1. Cree un archivo de manifiesto para el conjunto de datos de entrenamiento de la siguiente manera:

   - Cree un archivo de manifiesto con un SageMaker AI GroundTruth Job siguiendo las instrucciones que se indican en Etiquetar imágenes con un trabajo de Amazon SageMaker AI Ground Truth.

- Cree su propio archivo de manifiesto siguiendo las instrucciones de Creación de un archivo de manifiesto.

Si quiere crear un conjunto de datos de prueba, repita el paso 1 para crear el conjunto de datos de prueba.

2. Abra la consola Amazon Rekognition en. https://console.aws.amazon.com/rekognition/
3. Elija Usar etiquetas personalizadas.
4. Elija Comenzar.
5. En el panel de navegación izquierdo, elija Proyectos.
6. En la página Proyectos, elija el proyecto al que desee añadir el conjunto de datos. Se abrirá la página de detalles del proyecto.
7. Elija Crear conjunto de datos. Se abrirá la página Crear conjunto de datos.
8. En Configuración inicial, seleccione Empezar con un único conjunto de datos o Empezar con un conjunto de datos de entrenamiento. Para crear un modelo de mayor calidad, le recomendamos empezar con conjuntos de datos de entrenamiento y de prueba independientes.

   Single dataset

   a. En la sección de detalles del conjunto de datos de entrenamiento, selecciona Importar imágenes etiquetadas por SageMaker Ground Truth.
   b. En la ubicación del archivo de manifiesto, introduzca la ubicación del archivo de manifiesto que creó en el paso 1.
   c. Elija Crear conjunto de datos. Se abrirá la página de los conjuntos de datos de su proyecto.

   Separate training and test datasets

   a. En la sección de detalles del conjunto de datos de entrenamiento, selecciona Importar imágenes etiquetadas por SageMaker Ground Truth.
   b. En la ubicación del archivo de manifiesto, introduzca la ubicación del archivo de manifiesto del conjunto de datos de entrenamiento que creó en el paso 1.
   c. En la sección Detalles del conjunto de datos de prueba, selecciona Importar imágenes etiquetadas por SageMaker Ground Truth.

> **Note**
>
> Los conjuntos de datos de entrenamiento y de prueba pueden tener diferentes fuentes de imágenes.

   d. En la ubicación del archivo de manifiesto, introduzca la ubicación del archivo de manifiesto del conjunto de datos de prueba que creó en el paso 1.
   e. Elija Crear conjuntos de datos. Se abrirá la página de los conjuntos de datos de su proyecto.
9. Si necesita agregar o cambiar etiquetas, consulte Etiquetado de imágenes.
10. Siga los pasos que se indican en Entrenamiento de un modelo (consola) para entrenar el modelo.

Creación de un conjunto de datos con un archivo de manifiesto (SDK) de SageMaker AI Ground Truth

El siguiente procedimiento muestra cómo crear conjuntos de datos de entrenamiento o prueba a partir de un archivo de manifiesto mediante la CreateDatasetAPI.

Puedes usar un archivo de manifiesto existente, como el resultado de un trabajo de SageMaker AI Ground Truth, o crear tu propio archivo de manifiesto.

1. Si aún no lo has hecho, instala y configura el AWS CLI y el AWS SDKs. Para obtener más información, consulte Paso 4: Configure y AWS CLIAWS SDKs.
2. Cree un archivo de manifiesto para el conjunto de datos de entrenamiento de la siguiente manera:

   - Cree un archivo de manifiesto con un SageMaker AI GroundTruth Job siguiendo las instrucciones que se indican en Etiquetar imágenes con un trabajo de Amazon SageMaker AI Ground Truth.
   - Cree su propio archivo de manifiesto siguiendo las instrucciones de Creación de un archivo de manifiesto.

   Si quiere crear un conjunto de datos de prueba, repita el paso 2 para crear el conjunto de datos de prueba.
3. Use el siguiente código de ejemplo para crear el conjunto de datos de entrenamiento y prueba.

AWS CLI

Cree un conjunto de datos con el siguiente código. Sustituya lo siguiente:

- `project_arn`: el ARN del proyecto al que desea agregar el conjunto de datos de prueba.
- `type`: el tipo de conjunto de datos que desea crear (ENTRENAMIENTO o PRUEBA).
- `bucket`: el bucket que contiene el archivo de manifiesto del conjunto de datos.
- `manifest_file`: el nombre de archivo y la ruta del archivo de manifiesto.

```
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

Cree un conjunto de datos con los siguientes valores. Indique los siguientes parámetros de línea de comandos:

- `project_arn`: el ARN del proyecto al que desea agregar el conjunto de datos de prueba.
- `dataset_type`: el tipo de conjunto de datos que desea crear (`train` o `test`).
- `bucket`: el bucket que contiene el archivo de manifiesto del conjunto de datos.
- `manifest_file`: el nombre de archivo y la ruta del archivo de manifiesto.

```
#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

Cree un conjunto de datos con los siguientes valores. Indique los siguientes parámetros de línea de comandos:

- `project_arn`: el ARN del proyecto al que desea agregar el conjunto de datos de prueba.
- `dataset_type`: el tipo de conjunto de datos que desea crear (`train` o `test`).
- `bucket`: el bucket que contiene el archivo de manifiesto del conjunto de datos.
- `manifest_file`: el nombre de archivo y la ruta del archivo de manifiesto.

```
/*
   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. Si necesita agregar o cambiar etiquetas, consulte Administración de etiquetas (SDK).
5. Siga los pasos que se indican en Entrenamiento de un modelo (SDK) para entrenar el modelo.

Crear una solicitud de conjunto de datos

El formato de la solicitud de CreateDataset operación es el siguiente:

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

Etiquetar imágenes con un trabajo de Amazon SageMaker AI Ground Truth

Con Amazon SageMaker AI Ground Truth, puede utilizar trabajadores de Amazon Mechanical Turk, una empresa proveedora que elija, o de una fuerza laboral interna y privada, junto con el aprendizaje automático que le permite crear un conjunto de imágenes etiquetadas. Amazon Rekognition Custom Labels importa los archivos de manifiesto de SageMaker AI Ground Truth desde un bucket de Amazon S3 que especifique.

Las etiquetas personalizadas Amazon Rekognition son compatibles con las siguientes tareas de SageMaker AI Ground Truth.

- Clasificación de imágenes
- Cuadro delimitador

Los archivos que importe son las imágenes y un archivo de manifiesto. El archivo de manifiesto contiene información sobre las etiquetas y los cuadros delimitadores de las imágenes que importe.

Amazon Rekognition necesita permisos para acceder al bucket de Amazon S3 donde se almacenan las imágenes. Si utiliza el bucket de consola configurado por Etiquetas personalizadas de Amazon

Rekognition, los permisos necesarios ya estarán configurados. Si no utiliza el bucket de consola, consulte Acceso a buckets de Amazon S3 externos.

Creación de un archivo de manifiesto con un trabajo de SageMaker AI Ground Truth (consola)

El siguiente procedimiento muestra cómo crear un conjunto de datos mediante imágenes etiquetadas por un trabajo de SageMaker AI Ground Truth. Los archivos finales de los trabajos se almacenan en el bucket de consola de Etiquetas personalizadas de Amazon Rekognition.

Para crear un conjunto de datos con imágenes etiquetadas por un trabajo de SageMaker AI Ground Truth (consola)

1. Inicie sesión en la consola de Amazon S3 AWS Management Console y ábrala en https://console.aws.amazon.com/s3/.
2. En el bucket de consola, cree una carpeta para guardar las imágenes de entrenamiento.

   > **Note**
   >
   > El bucket de consola se crea al abrir por primera vez la consola Amazon Rekognition Custom Labels en una región. AWS Para obtener más información, consulte Administración de un proyecto de Etiquetas personalizadas de Amazon Rekognition.

3. Cargue sus imágenes en la carpeta que acaba de crear.
4. En el bucket de consola, cree una carpeta para guardar el resultado del trabajo de Ground Truth.
5. Abra la consola de SageMaker IA en. https://console.aws.amazon.com/sagemaker/
6. Cree un trabajo de etiquetado de Ground Truth. Necesitará Amazon S3 URLs para las carpetas que creó en los pasos 2 y 4. Para obtener más información, consulta Cómo usar Amazon SageMaker Ground Truth para el etiquetado de datos.
7. Anote la ubicación del archivo `output.manifest` en la carpeta que creó en el paso 4. Debe estar en la subcarpeta *`Ground-Truth-Job-Name`*`/manifests/output`.
8. Siga las instrucciones que aparecen en Creación de un conjunto de datos con un archivo de manifiesto de SageMaker AI Ground Truth (consola) para crear un conjunto de datos con el archivo de manifiesto cargado. Para el paso 8, en la ubicación del archivo de manifiesto, introduzca la URL de Amazon S3 para la ubicación que anotó en el paso anterior. Si está utilizando el AWS SDK, hágaloCreación de un conjunto de datos con un archivo de manifiesto (SDK) de SageMaker AI Ground Truth.

9. Repite los pasos 1 a 6 para crear el trabajo SageMaker AI Ground Truth para tu conjunto de datos de prueba.

Creación de un archivo de manifiesto

Puedes crear un conjunto de datos de prueba o entrenamiento importando un archivo de manifiesto en formato SageMaker AI Ground Truth. Si tus imágenes están etiquetadas en un formato que no es un archivo de manifiesto de SageMaker AI Ground Truth, usa la siguiente información para crear un archivo de manifiesto en formato SageMaker AI Ground Truth.

Los archivos de manifiesto están en formato de líneas JSON, donde cada línea es un objeto JSON completo que representa la información de etiquetado de una imagen. Las etiquetas personalizadas de Amazon Rekognition admiten los manifiestos de SageMaker AI Ground Truth con líneas JSON en los siguientes formatos:

- Resultado del trabajo de clasificación: sirve para agregar etiquetas de imagen a una imagen. Una etiqueta de imagen define la clase de escena, concepto u objeto (si no se necesita la información sobre la ubicación del objeto) que hay en una imagen. Una imagen puede tener más de una etiqueta de imagen. Para obtener más información, consulte Importación de etiquetas de imagen en los archivos de manifiesto.
- Resultado del trabajo del cuadro delimitador: sirve para etiquetar la clase y la ubicación de uno o varios objetos en una imagen. Para obtener más información, consulte Localización de objetos en archivos de manifiesto.

Las líneas JSON de imagen y de localización (cuadro delimitador) se pueden encadenar en el mismo archivo de manifiesto.

> **Note**
>
> Se ha cambiado el formato de los ejemplos de las líneas JSON de esta sección para facilitar su lectura.

Al importar un archivo de manifiesto, Etiquetas personalizadas de Amazon Rekognition aplica reglas de validación para los límites, la sintaxis y la semántica. Para obtener más información, consulte Reglas de validación de archivos de manifiesto.

Las imágenes a las que hace referencia un archivo de manifiesto deben estar ubicadas en el mismo bucket de Amazon S3. El archivo de manifiesto puede estar ubicado en un bucket de Amazon S3 distinto al que almacena las imágenes. Debe indicar la ubicación de una imagen en el campo `source-ref` de una línea JSON.

Amazon Rekognition necesita permisos para acceder al bucket de Amazon S3 donde se almacenan las imágenes. Si utiliza el bucket de consola configurado por Etiquetas personalizadas de Amazon Rekognition, los permisos necesarios ya estarán configurados. Si no utiliza el bucket de consola, consulte Acceso a buckets de Amazon S3 externos.

Temas

- Creación de un archivo de manifiesto

Creación de un archivo de manifiesto

Con el siguiente procedimiento se puede crear un proyecto con un conjunto de datos de entrenamiento y prueba. Los conjuntos de datos se crean a partir de los archivos de manifiesto de entrenamiento y prueba que se creen.

Para crear un conjunto de datos con un archivo de manifiesto en formato SageMaker AI Ground Truth (consola)

1. En el bucket de consola, cree una carpeta para guardar los archivos de manifiesto.
2. En el bucket de consola, cree una carpeta para guardar las imágenes.
3. Cargue su imágenes en la carpeta que acaba de crear.
4. Crea un archivo de manifiesto en formato SageMaker AI Ground Truth para tu conjunto de datos de entrenamiento. Para obtener más información, consulte Importación de etiquetas de imagen en los archivos de manifiesto y Localización de objetos en archivos de manifiesto.

   > **Important**
   >
   > El valor del campo `source-ref` de cada línea JSON debe asignarse a una imagen que haya cargado.

5. Crea un archivo de manifiesto en formato SageMaker AI Ground Truth para tu conjunto de datos de prueba.
6. Cargue los archivos de manifiesto en la carpeta que acaba de crear.

7. Anote la ubicación del archivo de manifiesto.
8. Siga las instrucciones que aparecen en Creación de un conjunto de datos con un archivo de manifiesto de SageMaker AI Ground Truth (consola) para crear un conjunto de datos con el archivo de manifiesto cargado. Para el paso 8, en la ubicación del archivo de manifiesto, introduzca la URL de Amazon S3 para la ubicación que anotó en el paso anterior. Si utilizas el AWS SDK, hazloCreación de un conjunto de datos con un archivo de manifiesto (SDK) de SageMaker AI Ground Truth.

Importación de etiquetas de imagen en los archivos de manifiesto

Para importar etiquetas a nivel de imagen (imágenes etiquetadas con escenas, conceptos u objetos que no requieren información de localización), añada líneas JSON en formato JSON en formato SageMaker AI Ground Truth Classification Job Output a un archivo de manifiesto. Un archivo de manifiesto está compuesto por una o varias líneas JSON, una para cada imagen que desee importar.

> **Tip**
>
> Para facilitar la creación de un archivo de manifiesto, habilitamos un script de Python que crea un archivo de manifiesto a partir de un archivo CSV. Para obtener más información, consulte Creación de un archivo de manifiesto a partir de un archivo CSV.

Cómo crear un archivo de manifiesto para etiquetas de imagen

1. Cree un archivo de texto vacío.
2. Añada una línea JSON por cada imagen que desee importar. La línea JSON debería tener un aspecto similar al siguiente.

```
{"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. Guarde el archivo. Puede usar la extensión `.manifest`, pero no es obligatoria.
4. Cree un conjunto de datos con el archivo de manifiesto que creó. Para obtener más información, consulte Para crear un conjunto de datos con un archivo de manifiesto en formato SageMaker AI Ground Truth (consola).

Líneas JSON de imagen

En esta sección, le indicamos cómo crear una línea JSON para una sola imagen. Analice la siguiente imagen. La escena para la siguiente imagen podría llamarse Amanecer.

La línea JSON de la imagen anterior, con la escena Amanecer, podría ser la siguiente.

```
{
    "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"
    }
}
```

Observe la siguiente información.

source-ref

(Obligatorio) La ubicación de Amazon S3 de la imagen. El formato es "s3://*BUCKET*/*OBJECT_PATH*". Las imágenes de un conjunto de datos importado deben almacenarse en el mismo bucket de Amazon S3.

testdataset-classification_Sunrise

(Obligatorio) El atributo de etiqueta. Elija el nombre del campo. El valor del campo (1 en el ejemplo anterior) es un identificador de atributo de etiqueta. No lo utilizan Etiquetas personalizadas de Amazon Rekognition y puede ser cualquier valor entero. Deben estar presentes los metadatos correspondientes identificados por el nombre del campo con el parámetro -metadata adjunto. Por ejemplo, "testdataset-classification_Sunrise-metadata".

testdataset-classification_Sunrise-metadatos

(Obligatorio) Metadatos sobre el atributo de etiqueta. El nombre del campo debe ser el mismo que el del atributo de etiqueta con -metadata anexado.

confidence

(Obligatorio) Etiquetas personalizadas de Amazon Rekognition no lo utiliza actualmente, pero se debe indicar un valor entre 0 y 1.

job-name

(Opcional) Un nombre que elija para el trabajo que procesa la imagen.

class-name

(Obligatorio) El nombre de clase que se elige para la escena o el concepto que se aplica a la imagen. Por ejemplo, "Sunrise".

human-annotated

(Obligatorio) Indique "yes" si la anotación la ha completado un humano. De lo contrario, "no".

creation-date

(Obligatorio) La fecha y la hora en que se creó la etiqueta, según la hora universal coordinada (UTC).

type

(Obligatorio) El tipo de procesamiento que se debe aplicar a la imagen. En el caso de las etiquetas de imagen, el valor es `"groundtruth/image-classification"`.

Cómo agregar varias etiquetas de imagen a una imagen

Puede agregar varias etiquetas a una imagen. Por ejemplo, el siguiente JSON agrega dos etiquetas, fútbol y pelota, a una sola imagen.

```
{
    "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
```

Localización de objetos en archivos de manifiesto

Puedes importar imágenes etiquetadas con información de localización de objetos añadiendo líneas JSON en formato SageMaker AI Ground Truth Bounding Box Job Output a un archivo de manifiesto.

La información de la localización representa la ubicación de un objeto en una imagen. La ubicación se representa mediante un cuadro delimitador que rodea el objeto. La estructura del cuadro delimitador incluye las coordenadas superiores izquierdas del cuadro delimitador y su ancho y alto. Una línea JSON con formato de cuadro delimitador incluye los cuadros delimitadores de las ubicaciones de uno o varios objetos en una imagen y la clase de cada objeto de la imagen.

Un archivo de manifiesto está compuesto por una o varias líneas JSON y cada línea contiene la información de una sola imagen.

Cómo crear un archivo de manifiesto para la localización de objetos

1. Cree un archivo de texto vacío.
2. Añada una línea JSON por cada imagen que desee importar. La línea JSON debería tener un aspecto similar al siguiente.

```
{"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. Guarde el archivo. Puede usar la extensión `.manifest`, pero no es obligatoria.
4. Cree un conjunto de datos con el archivo que creó. Para obtener más información, consulte Para crear un conjunto de datos con un archivo de manifiesto en formato SageMaker AI Ground Truth (consola).

Líneas JSON de cuadros delimitadores de objetos

En esta sección, le indicamos cómo crear una línea JSON para una sola imagen. Por ejemplo, en la siguiente imagen se ven unos cuadros delimitadores alrededor de un Amazon Echo y un Amazon Echo Dot.

La siguiente es la línea JSON del cuadro delimitador de la imagen anterior.


```
{
 "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"
  }
}
```

Observe la siguiente información.

source-ref

(Obligatorio) La ubicación de Amazon S3 de la imagen. El formato es "s3://*BUCKET*/*OBJECT_PATH*". Las imágenes de un conjunto de datos importado deben almacenarse en el mismo bucket de Amazon S3.

bounding-box

(Obligatorio) El atributo de etiqueta. Elija el nombre del campo. Incluye el tamaño de la imagen y los cuadros delimitadores de cada objeto detectado en la imagen. Deben estar presentes los metadatos correspondientes identificados por el nombre del campo con el parámetro -metadata adjunto. Por ejemplo, "bounding-box-metadata".

image_size

(Obligatorio) Una matriz de un solo elemento que contiene el tamaño de la imagen en píxeles.

- height: (obligatorio) la altura de la imagen en píxeles.
- width: (obligatorio) la profundidad de la imagen en píxeles.
- depth: (obligatorio) el número de canales de la imagen. En la caso de las imágenes RGB, el valor es 3. Etiquetas personalizadas de Amazon Rekognition no lo utiliza actualmente, pero se debe indicar un valor.

annotations

(Obligatorio) Un conjunto de información sobre los cuadros delimitadores de cada objeto detectado en la imagen.

- class_id: (obligatorio) se asigna a la etiqueta en class-map. En el ejemplo anterior, el objeto con el class_id de 1 es el Echo Dot de la imagen.
- top: (obligatorio) la distancia desde la parte superior de la imagen hasta la parte superior del cuadro delimitador, en píxeles.
- left: (obligatorio) la distancia desde la izquierda de la imagen hasta la izquierda del cuadro delimitador, en píxeles.
- width: (obligatorio) el ancho del cuadro delimitador, en píxeles.
- height: (obligatorio) la altura del cuadro delimitador, en píxeles.

bounding-box-metadatos

(Obligatorio) Metadatos sobre el atributo de etiqueta. El nombre del campo debe ser el mismo que el del atributo de etiqueta con el parámetro -metadata adjunto. Un conjunto de información sobre los cuadros delimitadores de cada objeto detectado en la imagen.

Objects

(Obligatorio) Una matriz de objetos presente en la imagen. Se asigna a la matriz de annotations por índice. Etiquetas personalizadas de Amazon Rekognition no utiliza el atributo confidence.

class-map

(Obligatorio) El mapa de clases que se aplica a los objetos detectados en la imagen.

type

(Obligatorio) El tipo de trabajo de clasificación. `"groundtruth/object-detection"` identifica el trabajo como detección de objetos.

creation-date

(Obligatorio) La fecha y la hora en que se creó la etiqueta, según la hora universal coordinada (UTC).

human-annotated

(Obligatorio) Indique `"yes"` si la anotación la ha completado un humano. De lo contrario, `"no"`.

job-name

(Opcional) El nombre del trabajo que procesa la imagen.

Reglas de validación de archivos de manifiesto

Al importar un archivo de manifiesto, Etiquetas personalizadas de Amazon Rekognition aplica reglas de validación para los límites, la sintaxis y la semántica. El esquema SageMaker AI Ground Truth impone la validación de la sintaxis. Para obtener más información, consulte Salida. Las siguientes son las reglas de validación de los límites y la semántica.

> **Note**
>
> - Las reglas de invalidez del 20 % se aplican de forma acumulativa a todas las reglas de validación. Si la importación supera el límite del 20 % debido a alguna combinación (por ejemplo, un 15 % de JSON no válido y un 15 % de imágenes no válidas), se producirá un error en la importación.
> - Cada objeto del conjunto de datos es una línea del manifiesto. Las líneas en blanco o no válidas también se cuentan como objetos del conjunto de datos.
> - Las superposiciones son (etiquetas comunes entre la prueba y el entrenamiento)/(etiquetas de entrenamiento).

Temas

- Límites
- Semántica

Límites

Validación	Límite	Se ha producido un error
Tamaño del archivo de manifiesto	1 GB máximo	Error
Número máximo de líneas para un archivo de manifiesto	Máximo de 250 000 objetos de conjunto de datos como líneas en un manifiesto.	Error
Límite inferior del número total de objetos de conjunto de datos válidos por etiqueta	>= 1	Error
Límite inferior de etiquetas	2	Error
Límite superior de etiquetas	<= 250	Error
Número mínimo de cuadros delimitadores por imagen	0	Ninguno
Número máximo de cuadros delimitadores por imagen	50	Ninguno

Semántica

Validación	Límite	Se ha producido un error
Manifiesto vacío		Error
Falta el objeto source-ref o no se puede acceder a él	Número de objetos inferior al 20 %	Advertencia
Falta el objeto source-ref o no se puede acceder a él	Número de objetos > 20 %	Error

Validación	Límite	Se ha producido un error
Las etiquetas de prueba no están presentes en el conjunto de datos de entrenamiento	Al menos un 50 % se superpone en las etiquetas	Error
Combinación de ejemplos de etiquetas y objetos para la misma etiqueta en un conjunto de datos. Clasificación y detección de la misma clase en un objeto de conjunto de datos.		Sin errores ni advertencias
Superposición de activos entre la prueba y el entrenamiento	No debe haber una superposición entre los conjuntos de datos de prueba y de entrenamiento.	
Las imágenes de un conjunto de datos deben proceder del mismo bucket	Se produce un error si los objetos están en un bucket diferente	Error

Convertir otros formatos de conjunto de datos en un archivo de manifiesto

Puede utilizar la siguiente información para crear archivos de manifiesto en formato Amazon SageMaker AI a partir de diversos formatos de conjuntos de datos de origen. Después de crear el archivo de manifiesto, úselo para crear un conjunto de datos. Para obtener más información, consulte Utilizar un archivo de manifiesto para importar imágenes.

Temas

- Transformar un conjunto de datos COCO en un formato de archivo de manifiesto
- Transformación de los archivos de manifiesto de SageMaker AI Ground Truth con múltiples etiquetas
- Creación de un archivo de manifiesto a partir de un archivo CSV

Transformar un conjunto de datos COCO en un formato de archivo de manifiesto

COCO es un formato que indica conjuntos de datos de detección, segmentación y subtitulación de objetos a gran escala. En este ejemplo de Python se explica cómo transformar un conjunto de datos con formato de detección de objetos COCO en un archivo de manifiesto con formato de cuadro delimitador de Etiquetas personalizadas de Amazon Rekognition. En esta sección también se incluye información que puede usar para escribir el código.

Un archivo JSON en formato COCO consta de cinco secciones que aportan información para un conjunto de datos completo. Para obtener más información, consulte El formato del conjunto de datos COCO.

- `info`: información general sobre el conjunto de datos.
- `licenses` : información de las licencias de imágenes del conjunto de datos.
- `images`: lista de imágenes del conjunto de datos.
- `annotations`: lista de anotaciones (incluidos los cuadros delimitadores) que están presentes en todas las imágenes del conjunto de datos.
- `categories`: lista de categorías de etiquetas.

Necesitará información de las listas `images`, `annotations` y `categories` para crear un archivo de manifiesto de Etiquetas personalizadas de Amazon Rekognition.

Un archivo de manifiesto de Etiquetas personalizadas de Amazon Rekognition tiene un formato de líneas JSON, donde cada línea tiene el cuadro delimitador y la información de etiqueta de uno o varios objetos de una imagen. Para obtener más información, consulte Localización de objetos en archivos de manifiesto.

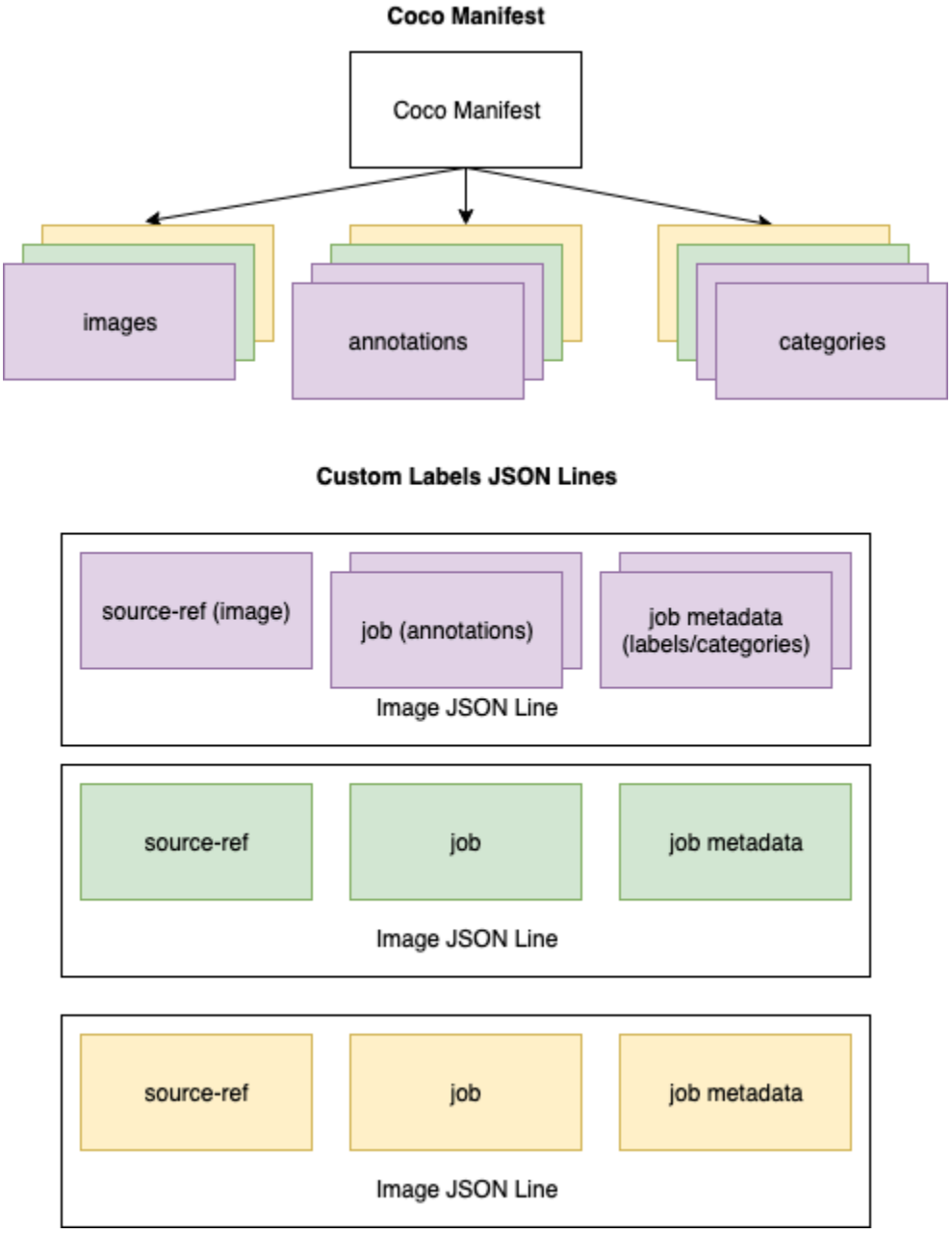
Asignación de objetos COCO a una línea JSON de etiquetas personalizadas

Para transformar un conjunto de datos en formato COCO, asigne el conjunto de datos COCO a un archivo de manifiesto de Etiquetas personalizadas de Amazon Rekognition para la localización de objetos. Para obtener más información, consulte Localización de objetos en archivos de manifiesto. Para crear una línea JSON para cada imagen, el archivo de manifiesto debe mapear el conjunto de datos `image` COCO y el campo `category` del objeto IDs. `annotation`

A continuación se muestra un ejemplo de un archivo de manifiesto COCO. Para obtener más información, consulte El formato del conjunto de datos COCO.

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

En el siguiente diagrama, se indica cómo las listas del conjunto de datos COCO para un conjunto de datos se asignan a las líneas JSON de Etiquetas personalizadas de Amazon Rekognition para una imagen. Cada línea JSON de una imagen contiene un campo de referencia de origen, trabajo y metadatos de trabajo. Los colores iguales señalan la información de una sola imagen. Tenga en cuenta que, en el manifiesto, una imagen individual puede tener varias anotaciones y metadatos/categorías.



Cómo obtener los objetos COCO de una sola línea JSON

1. En cada imagen de la lista de imágenes, obtenga la anotación de la lista de anotaciones en la que el valor del campo de anotación `image_id` coincida con el campo `id` de la imagen.
2. En cada anotación que coincida en el paso 1, revise la lista `categories` y obtenga cada `category` donde el valor `category` del campo `id` coincida con el campo `annotation` del objeto `category_id`.

3. Cree una línea JSON para la imagen utilizando los objetos `image`, `annotation` y `category` que coincidan. Para asignar los campos, consulte Asignación de campos de objetos COCO a campos de objeto de una línea JSON de etiquetas personalizadas.
4. Repita los pasos del 1 al 3 hasta que haya creado líneas JSON para cada objeto `image` de la lista `images`.

Para ver el código de ejemplo, consulte Transformación de un conjunto de datos COCO.

Asignación de campos de objetos COCO a campos de objeto de una línea JSON de etiquetas personalizadas

Tras identificar los objetos COCO de una línea JSON de Etiquetas personalizadas de Amazon Rekognition, debe asignar los campos de objetos COCO a los campos de objetos correspondientes de la línea JSON de Etiquetas personalizadas de Amazon Rekognition. En el siguiente ejemplo, la línea JSON de Etiquetas personalizadas de Amazon Rekognition asigna una línea de imagen (`id=000000245915`) al ejemplo anterior de JSON COCO. Observe la siguiente información.

- `source-ref` es la ubicación de la imagen en un bucket de Amazon S3. Si las imágenes COCO no están almacenadas en un bucket de Amazon S3, debe moverlas a un bucket de Amazon S3.
- La lista `annotations` contiene un objeto `annotation` por cada objeto de la imagen. Un objeto `annotation` incluye información sobre un cuadro delimitador (`top`, `left`, `width`, `height`) y un identificador de etiqueta (`class_id`).
- El identificador de etiqueta (`class_id`) se asigna a la lista `class-map` en los metadatos. Aquí aparecen las etiquetas utilizadas en la imagen.

```
{
 "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"
  }
}
```

Utilice la siguiente información para asignar los campos del archivo de manifiesto de Etiquetas personalizadas de Amazon Rekognition a los campos JSON del conjunto de datos COCO.

source-ref

La URL en formato S3 para la ubicación de la imagen. La imagen vídeo debe almacenarse en un bucket de S3. Para obtener más información, consulte source-ref. Si el campo COCO `coco_url` enlaza con una ubicación del bucket de S3, puede usar el valor de `coco_url` para el valor de `source-ref`. Como alternativa, puede asignar `source-ref` al campo `file_name` (COCO) y, en el código de transformación, agregar la ruta de S3 necesaria a la ubicación donde está almacenada la imagen.

bounding-box

El nombre de atributo de etiqueta que elija. Para obtener más información, consulte *bounding-box*.

image_size

Tamaño de la imagen en píxeles. Se asigna a un objeto `image` de la lista de imágenes.

- `height` -> `image.height`
- `width` -> `image.width`
- `depth` -> Etiquetas personalizadas de Amazon Rekognition no lo utiliza, pero se debe indicar un valor.

annotations

Una lista de objetos `annotation`. Hay un `annotation` por cada objeto de la imagen.

anotación

Contiene información sobre un cuadro delimitador para la instancia de un objeto de la imagen.

- `class_id` -> Asignación de identificadores numéricos a la lista `class-map` de etiquetas personalizadas.
- `top` -> `bbox[1]`
- `left` -> `bbox[0]`
- `width` -> `bbox[2]`
- `height` -> `bbox[3]`

bounding-box-metadatos

Metadatos del atributo de etiqueta. Incluye las etiquetas y los identificadores de etiquetas. Para obtener más información, consulte *bounding-box*-metadatos.

Objects

Una matriz de objetos de la imagen. Asigna la lista `annotations` por índice.

Objeto

- `confidence` ->Etiquetas personalizadas de Amazon Rekognition no lo utiliza, pero es obligatorio indicar un valor (1).

class-map

Un mapa de clases (classes) que se aplican a los objetos detectados en la imagen. Se asigna a los objetos de categoría de la lista de categorías.

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

type

Debe ser `groundtruth/object-detection`

human-annotated

Indique `yes` o `no`. Para obtener más información, consulte *`bounding-box`*-metadatos.

creation-date -> image.date_captured

La fecha y hora de creación de la imagen. Se asigna al campo image .date_captured de una imagen de la lista de imágenes COCO. Etiquetas personalizadas de Amazon Rekognition espera que el formato de `creation-date` sea Y-M-DTH:M:S.

job-name

El nombre del trabajo que prefiera.

El formato del conjunto de datos COCO

Un conjunto de datos COCO consta de cinco secciones de información que aportan información para todo el conjunto de datos. El formato de un conjunto de datos de detección de objetos COCO viene documentado en Formato de datos COCO.

- info: información general sobre el conjunto de datos.
- licenses: información de las licencias de imágenes del conjunto de datos.
- images: lista de imágenes del conjunto de datos.
- annotations: lista de anotaciones (incluidos los cuadros delimitadores) que están presentes en todas las imágenes del conjunto de datos.
- categories: lista de categorías de etiquetas.

Para crear un manifiesto de etiquetas personalizadas, utilice las listas `images`, `annotations` y `categories` del archivo de manifiesto COCO. Las demás secciones (`info`, `licences`) no son obligatorias. A continuación se muestra un ejemplo de un archivo de manifiesto COCO.

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

images list

Las imágenes a las que hace referencia un conjunto de datos COCO figuran en la matriz de imágenes. Cada objeto de imagen contiene información sobre la imagen, como el nombre de archivo de la imagen. En el siguiente objeto de imagen de ejemplo, fíjese en la siguiente información y los campos necesarios para crear un archivo de manifiesto de Etiquetas personalizadas de Amazon Rekognition.

- `id`: (Obligatorio) Un identificador único para la imagen. El campo `id` se asigna al campo `id` de la matriz de anotaciones (donde se almacena la información del cuadro delimitador).
- `license`: (No obligatorio) Se asigna a la matriz de licencias.
- `coco_url`: (Opcional) La ubicación de la imagen.
- `flickr_url`: (No obligatorio) La ubicación de la imagen en Flickr.
- `width`: (Obligatorio) El ancho de la imagen.
- `height`: (Obligatorio) La altura de la imagen.
- `file_name`: (Obligatorio) El nombre del archivo de imagen. En este ejemplo, `file_name` y `id` coinciden, pero esto no es un requisito para los conjuntos de datos COCO.
- `date_captured`: (Obligatorio) La fecha y la hora en que se capturó la imagen.

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

lista de anotaciones (cuadros delimitadores)

La información de los cuadros delimitadores de todos los objetos en todas las imágenes se almacena en la lista de anotaciones. Un único objeto de anotación contiene la información del cuadro delimitador de un único objeto y la etiqueta del objeto en una imagen. Hay un objeto de anotación por cada instancia de un objeto en una imagen.

En el siguiente ejemplo, fíjese en la siguiente información y los campos necesarios para crear un archivo de manifiesto de Etiquetas personalizadas de Amazon Rekognition.

- `id`: (No obligatorio) El identificador de la anotación.
- `image_id`: (Obligatorio) Corresponde a la imagen `id` en la matriz de imágenes.
- `category_id`: (Obligatorio) El identificador de la etiqueta que identifica el objeto dentro de un cuadro delimitador. Se asigna al campo `id` de la matriz de categorías.
- `iscrowd`: (No obligatorio) Señala si la imagen tiene muchos objetos.
- `segmentation`: (No obligatorio) Información de segmentación de los objetos de una imagen. Etiquetas personalizadas de Amazon Rekognition no admite la segmentación.
- `area`: (No obligatorio) El área de la anotación.
- `bbox`: (Obligatorio) Incluye las coordenadas, en píxeles, de un cuadro delimitador alrededor de un objeto de la imagen.

```
{
    "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]
}
```

lista de categorías

La información de las etiquetas se almacena en la matriz de categorías. En el siguiente objeto de categoría de ejemplo, fíjese en la siguiente información y los campos necesarios para crear un archivo de manifiesto de Etiquetas personalizadas de Amazon Rekognition.

- `supercategory`: (No obligatorio) La categoría principal de una etiqueta.
- `id`: (Obligatorio) El identificador de la etiqueta. El campo `id` se asigna al campo `category_id` de un objeto `annotation`. En el siguiente ejemplo, el identificador de un echo dot es 2.
- `name`: (Obligatorio) El nombre de la etiqueta.

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

Transformación de un conjunto de datos COCO

Use el siguiente ejemplo de Python para transformar la información de un cuadro delimitador de un conjunto de datos con formato COCO en un archivo de manifiesto de Etiquetas personalizadas de Amazon Rekognition. El código carga el archivo de manifiesto creado en el bucket de Amazon S3. El código también incluye un comando de AWS CLI que puede utilizar para cargar las imágenes.

Cómo transformar un conjunto de datos COCO (SDK)

1. Si aún no lo ha hecho:

   a. Asegúrese de que tiene los permisos de `AmazonS3FullAccess`. Para obtener más información, consulte Configuración de permisos de SDK.

   b. Instale y configure el AWS CLI y el AWS SDKs. Para obtener más información, consulte Paso 4: Configure y AWS CLIAWS SDKs.

2. Usa el siguiente código de Python para transformar un conjunto de datos COCO. Defina los siguientes valores.

   - `s3_bucket`: el nombre del bucket de S3 en el que desea almacenar las imágenes y el archivo de manifiesto de Etiquetas personalizadas de Amazon Rekognition.
   - `s3_key_path_images`: la ruta en la que desea colocar las imágenes dentro del bucket de S3 (`s3_bucket`).
   - `s3_key_path_manifest_file`: la ruta en la que desea colocar el archivo de manifiesto de Etiquetas personalizadas dentro del bucket de S3 (`s3_bucket`).
   - `local_path`: la ruta local donde el ejemplo abre el conjunto de datos COCO de entrada y también guarda el nuevo archivo de manifiesto de Etiquetas personalizadas.
   - `local_images_path`: la ruta local de las imágenes que quiera usar para el entrenamiento.
   - `coco_manifest`: el nombre de archivo del conjunto de datos COCO de entrada.
   - `cl_manifest_file`: el nombre del archivo de manifiesto creado en el ejemplo. El archivo se guarda en la ubicación indicada por `local_path`. Por convención, el archivo tiene la extensión `.manifest`, pero no es obligatoria.
   - `job_name`: el nombre para el trabajo de Etiquetas personalizadas.

```
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. Ejecute el código.
4. En el resultado del programa, anote el comando `s3 sync`. Lo necesitará en el siguiente paso.
5. En el símbolo del sistema, ejecute el comando `s3 sync`. Las imágenes se cargan en el bucket de S3. Si el comando falla durante la carga, ejecútelo de nuevo hasta que las imágenes locales se sincronicen con el bucket de S3.
6. En el resultado del programa, anote la ruta URL de S3 correspondiente al archivo de manifiesto. Lo necesitará en el siguiente paso.
7. Siga las instrucciones que aparecen en Creación de un conjunto de datos con un archivo de manifiesto de SageMaker AI Ground Truth (consola) para crear un conjunto de datos con el archivo de manifiesto cargado. En el paso 8, en la ubicación del archivo de manifiesto, introduzca la URL de Amazon S3 que haya anotado en el paso anterior. Si utiliza el AWS SDK, consulte Creación de un conjunto de datos con un archivo de manifiesto (SDK) de SageMaker AI Ground Truth.

Transformación de los archivos de manifiesto de SageMaker AI Ground Truth con múltiples etiquetas

En este tema se muestra cómo transformar un archivo de manifiesto de Amazon SageMaker AI Ground Truth con varias etiquetas en un archivo de manifiesto con formato Amazon Rekognition Custom Labels.

SageMaker Los archivos de manifiesto de AI Ground Truth para trabajos con varias etiquetas tienen un formato diferente al de los archivos de manifiesto con formato Amazon Rekognition Custom Labels. La clasificación de etiquetas múltiples consiste en clasificar una imagen en un conjunto de clases, aunque puede pertenecer a varias clases a la vez. En este caso, es posible que la imagen tenga varias etiquetas (etiquetas múltiples), como fútbol y pelota.

Para obtener información sobre los trabajos de SageMaker AI Ground Truth con múltiples etiquetas, consulte Clasificación de imágenes (etiquetas múltiples). Para obtener información sobre los archivos de manifiesto de etiquetas múltiples de Etiquetas personalizadas de Amazon Rekognition, consulte the section called “Cómo agregar varias etiquetas de imagen a una imagen”.

Obtener el archivo de manifiesto para un trabajo de SageMaker AI Ground Truth

El siguiente procedimiento muestra cómo obtener el archivo de manifiesto de salida (`output.manifest`) para un trabajo de Amazon SageMaker AI Ground Truth. Usará `output.manifest` como entrada en el procedimiento siguiente.

Para descargar un archivo de manifiesto de trabajo de SageMaker AI Ground Truth

1. Abra la https://console.aws.amazon.com/sagemaker/.
2. En el panel de navegación, elija Ground Truth y luego Trabajos de etiquetado.
3. Elija el trabajo de etiquetado que contenga el archivo de manifiesto que desee usar.
4. En la página de detalles, elija el enlace situado debajo de la ubicación del conjunto de datos final. Se abrirá la consola de Amazon S3 en la ubicación del conjunto de datos.
5. Elija `Manifests`, `output` y luego `output.manifest`.
6. Elija Acciones de objeto y después Descargar para descargar el archivo de manifiesto.

Transformar un archivo de manifiesto de SageMaker IA con varias etiquetas

El siguiente procedimiento crea un archivo de manifiesto Amazon Rekognition Custom Labels con formato de etiquetas múltiples a partir de un archivo de manifiesto AI con formato multietiqueta existente. SageMaker GroundTruth

> **Note**
>
> Para ejecutar el código, necesita Python, versión 3 o superior.

Para transformar un archivo de manifiesto de IA con varias etiquetas SageMaker

1. Ejecute el siguiente código de Python. Indique el nombre del archivo de manifiesto que haya creado en Obtener el archivo de manifiesto para un trabajo de SageMaker AI Ground Truth como argumento de línea de comandos.

```
# 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. Anote el nombre del nuevo archivo de manifiesto que aparece en el script. Lo usará en el siguiente paso.
3. Cargue los archivos de manifiesto en el bucket de Amazon S3 que desee utilizar para almacenar el archivo de manifiesto.

> **Note**
>
> Asegúrese de que Etiquetas personalizadas de Amazon Rekognition tenga acceso al bucket de Amazon S3 al que se hace referencia en el campo `source-ref` de las líneas JSON del archivo de manifiesto. Para obtener más información, consulte Acceso a buckets de Amazon S3 externos. Si el trabajo en Ground Truth almacena imágenes en el bucket de consola de Etiquetas personalizadas de Amazon Rekognition, no necesitará agregar permisos.

4. Siga las instrucciones que aparecen en Creación de un conjunto de datos con un archivo de manifiesto de SageMaker AI Ground Truth (consola) para crear un conjunto de datos con el archivo de manifiesto cargado. En el paso 8, en la ubicación del archivo de manifiesto, introduzca la URL de Amazon S3 de la ubicación que haya anotado en el paso anterior. Si utiliza

el AWS SDK, consulte Creación de un conjunto de datos con un archivo de manifiesto (SDK) de SageMaker AI Ground Truth.

Creación de un archivo de manifiesto a partir de un archivo CSV

Este script de Python de ejemplo facilita la creación de un archivo de manifiesto mediante el uso de un archivo de valores separados por comas (CSV) para etiquetar las imágenes. Usted crea el archivo CSV. El archivo de manifiesto se puede usar para la clasificación de imágenes de etiquetas múltiples o Clasificación de imágenes de etiquetas múltiples. Para obtener más información, consulte Buscar objetos, escenas y conceptos.

> **Note**
>
> Este script no crea un archivo de manifiesto adecuado para buscar ubicaciones de objetos o ubicaciones de marcas.

El archivo de manifiesto describe las imágenes utilizadas para entrenar un modelo. Por ejemplo, las ubicaciones de las imágenes y las etiquetas asignadas a las imágenes. Un archivo de manifiesto se compone de una o varias líneas JSON. Cada línea JSON describe una sola imagen. Para obtener más información, consulte the section called “Importación de etiquetas de imagen en los archivos de manifiesto”.

Un archivo CSV representa datos tabulares en varias filas de un archivo de texto. Los campos de las filas están separados por comas. Para obtener más información, consulte valores separados por comas. En este script, cada fila del archivo CSV representa una sola imagen y se asigna a una línea JSON del archivo de manifiesto. Para crear el archivo CSV de un archivo de manifiesto que admita la clasificación de imágenes de etiquetas múltiples, agregue una o varias etiquetas de imagen a cada fila. Para crear un archivo de manifiesto adecuado para Clasificación de imágenes, agregue una sola etiqueta de imagen a cada fila.

Por ejemplo, el siguiente archivo CSV describe las imágenes del proyecto Clasificación de imágenes de etiquetas múltiples (flores) de Introducción.

```
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
```

El script genera líneas JSON para cada fila. Por ejemplo, la siguiente es la línea JSON de la primera fila (`camellia1.jpg,camellia,with_leaves`).

```
{"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"}}
```

En el CSV de ejemplo, la ruta de Amazon S3 de la imagen no está presente. Si el archivo CSV no incluye la ruta de Amazon S3 de las imágenes, utilice el argumento `--s3_path` de la línea de comandos para indicar la ruta de Amazon S3 correspondiente a la imagen.

El script registra la primera entrada de cada imagen en un archivo CSV de imágenes deduplicadas. El archivo CSV de imágenes deduplicadas contiene una sola instancia por cada imagen que se encuentra en el archivo CSV de entrada. Las demás apariciones de una imagen en el archivo CSV de entrada se registra en un archivo CSV de imágenes duplicadas. Si el script encuentra imágenes duplicadas, revise el archivo CSV de imágenes duplicadas y actualice el archivo CSV de imágenes deduplicadas según sea necesario. Vuelva a ejecutar el script con el archivo deduplicado. Si no se encuentra ningún duplicado en el archivo CSV de entrada, el script elimina el archivo CSV de la imagen deduplicada y la imagen duplicada CSVfile, ya que están vacíos.

En este procedimiento, se crea el archivo CSV y se ejecuta el script de Python para crear el archivo de manifiesto.

Cómo crear un archivo de manifiesto a partir de un archivo CSV

1. Cree un archivo CSV con los siguientes campos en cada fila (una fila por imagen). No añadas una fila de encabezado al archivo CSV.

Campo 1	Campo 2	Campo n
El nombre de la imagen o la ruta de Amazon S3 de la imagen. Por ejemplo, `s3://my-bucket/flowers/train/camellia1.jpg` . No se pueden mezclar imágenes con la ruta de Amazon S3 e imágenes sin dicha ruta.	La primera etiqueta de imagen de la imagen.	Una o varias etiquetas adicionales de imagen separadas por comas. Añádalas solo si desea crear un archivo de manifiesto que admita la clasificación de imágenes con etiquetas múltiples.

   Por ejemplo, `camellia1.jpg,camellia,with_leaves` o `s3://my-bucket/flowers/train/camellia1.jpg,camellia,with_leaves`
2. Guarde el archivo CSV.
3. Ejecute el siguiente script de Python. Proporcione los siguientes argumentos:
   - `csv_file`: el archivo CSV creado en el paso 5.
   - `manifest_file`: El nombre del archivo de manifiesto que desea crear.
   - (Opcional)`--s3_path` *`s3://path_to_folder/`*: la ruta de Amazon S3 que se quiera agregar de los archivos de imagen (campo 1). Use `--s3_path` si las imágenes del campo 1 aún no contienen una ruta de S3.

```
# 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. Si tiene pensado usar un conjunto de datos de prueba, repita los pasos del 1 al 3 para crear un archivo de manifiesto para su conjunto de datos de prueba.
5. Si es necesario, copie las imágenes en la ruta del bucket de Amazon S3 que haya indicado en la columna 1 del archivo CSV (o que haya indicado en la línea de comandos `--s3_path`). Puede utilizar el siguiente comando de AWS S3.

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

6. Cargue los archivos de manifiesto en el bucket de Amazon S3 que desee utilizar para almacenar el archivo de manifiesto.

> **Note**
>
> Asegúrese de que Etiquetas personalizadas de Amazon Rekognition tenga acceso al bucket de Amazon S3 al que se hace referencia en el campo `source-ref` de las líneas JSON del archivo de manifiesto. Para obtener más información, consulte Acceso a buckets de Amazon S3 externos. Si el trabajo en Ground Truth almacena imágenes en

el bucket de consola de Etiquetas personalizadas de Amazon Rekognition, no necesitará agregar permisos.

7. Siga las instrucciones que aparecen en Creación de un conjunto de datos con un archivo de manifiesto de SageMaker AI Ground Truth (consola) para crear un conjunto de datos con el archivo de manifiesto cargado. En el paso 8, en la ubicación del archivo de manifiesto, introduzca la URL de Amazon S3 de la ubicación que haya anotado en el paso anterior. Si utiliza el AWS SDK, consulte Creación de un conjunto de datos con un archivo de manifiesto (SDK) de SageMaker AI Ground Truth.

Copia de contenido de un conjunto de datos existente

Si ya ha creado un conjunto de datos, puede copiar su contenido en un conjunto de datos nuevo. Para crear un conjunto de datos a partir de un conjunto de datos existente con el AWS SDK, consulte. Creación de un conjunto de datos mediante un conjunto de datos existente (SDK)

Cómo crear un conjunto de datos mediante un conjunto de datos existente de Etiquetas personalizadas de Amazon Rekognition existente (consola)

1. Abra la consola Amazon Rekognition en. https://console.aws.amazon.com/rekognition/
2. Elija Usar etiquetas personalizadas.
3. Elija Comenzar.
4. En el panel de navegación izquierdo, elija Proyectos.
5. En la página Proyectos, elija el proyecto al que desee añadir el conjunto de datos. Se abrirá la página de detalles del proyecto.
6. Elija Crear conjunto de datos. Se abrirá la página Crear conjunto de datos.
7. En Configuración inicial, seleccione Empezar con un único conjunto de datos o Empezar con un conjunto de datos de entrenamiento. Para crear un modelo de mayor calidad, le recomendamos empezar con conjuntos de datos de entrenamiento y de prueba independientes.

   Single dataset

   a. En la sección Detalles del conjunto de datos de entrenamiento, seleccione Copiar un conjunto de datos de Etiquetas personalizadas de Amazon Rekognition existente.

b. En la sección Detalles del conjunto de datos de entrenamiento, en el cuadro de edición Conjunto de datos, escriba o seleccione el nombre del conjunto de datos que desee copiar.

c. Elija Crear conjunto de datos. Se abrirá la página de los conjuntos de datos de su proyecto.

Separate training and test datasets

a. En la sección Detalles del conjunto de datos de entrenamiento, seleccione Copiar un conjunto de datos de Etiquetas personalizadas de Amazon Rekognition existente.

b. En la sección Detalles del conjunto de datos de entrenamiento, en el cuadro de edición Conjunto de datos, escriba o seleccione el nombre del conjunto de datos que desee copiar.

c. En la sección Detalles del conjunto de datos de prueba, seleccione Copiar un conjunto de datos de Etiquetas personalizadas de Amazon Rekognition existente.

d. En la sección Detalles del conjunto de datos de prueba, en el cuadro de edición Conjunto de datos, escriba o seleccione el nombre del conjunto de datos que desee copiar.

> **Note**
>
> Los conjuntos de datos de entrenamiento y de prueba pueden tener diferentes fuentes de imágenes.

e. Elija Crear conjuntos de datos. Se abrirá la página de los conjuntos de datos de su proyecto.

8. Si necesita agregar o cambiar etiquetas, consulte Etiquetado de imágenes.
9. Siga los pasos que se indican en Entrenamiento de un modelo (consola) para entrenar el modelo.

Etiquetado de imágenes

Una etiqueta identifica un objeto, una escena, un concepto o un cuadro delimitador alrededor del objeto de una imagen. Por ejemplo, si el conjunto de datos contiene imágenes de perros, puede agregar etiquetas de razas de perros.

Después de importar las imágenes a un conjunto de datos, es posible que tenga que agregar etiquetas a las imágenes o corregir las imágenes mal etiquetadas. Por ejemplo, las imágenes importadas de un ordenador local no vienen etiquetadas. Utilice la galería de conjuntos de datos para agregar nuevas etiquetas al conjunto de datos y asignar etiquetas y cuadros delimitadores a las imágenes del conjunto de datos.

La forma en que etiquete las imágenes en sus conjuntos de datos determina el tipo de modelo que entrena Etiquetas personalizadas de Amazon Rekognition. Para obtener más información, consulte Finalidad de los conjuntos de datos.

Temas

- Administración de etiquetas
- Asignación de etiquetas de imagen a una imagen
- Etiquetado de objetos con cuadros delimitadores

Administración de etiquetas

Puede gestionar las etiquetas mediante la consola de Etiquetas personalizadas de Amazon Rekognition. No hay una API específica para administrar las etiquetas: las etiquetas se agregan al conjunto de datos cuando se crea el conjunto de datos con `CreateDataset` o cuando se agregan más imágenes al conjunto de datos con `UpdateDatasetEntries`.

Temas

- Administración de etiquetas (consola)
- Administración de etiquetas (SDK)

Administración de etiquetas (consola)

Puede utilizar la consola de Etiquetas personalizadas de Amazon Rekognition para agregar, cambiar o eliminar etiquetas. Para agregar una etiqueta a un conjunto de datos, puede agregar una etiqueta nueva que cree o importar etiquetas de un conjunto de datos existente en Rekognition.

Temas

- Agregar etiquetas nuevas (consola)
- Cambiar y eliminar etiquetas (consola)

Agregar etiquetas nuevas (consola)

Puede indicar nuevas etiquetas que desee agregar al conjunto de datos.

Agregar etiquetas mediante la ventana de edición

Cómo agregar una nueva etiqueta (consola)

1. Abra la consola Amazon Rekognition en. https://console.aws.amazon.com/rekognition/
2. Elija Usar etiquetas personalizadas.
3. Elija Comenzar.
4. En el panel de navegación izquierdo, elija Proyectos.
5. En la página Proyectos, elija el proyecto que desee usar. Se abrirá la página de detalles del proyecto.
6. Si quiere agregar etiquetas al conjunto de datos de entrenamiento, elija la pestaña Entrenamiento. Si no, seleccione la pestaña Prueba para agregar etiquetas al conjunto de datos de prueba.
7. Elija Comenzar a etiquetar para activar el modo de etiquetado.
8. En la sección Etiquetas de la galería de conjuntos de datos, elija Administrar etiquetas para abrir el cuadro de diálogo Administrar etiquetas.
9. En el cuadro de edición, introduzca un nombre de etiqueta nuevo.
10. Elija Agregar nueva etiqueta.
11. Repita los pasos 9 y 10 hasta que haya creado todas las etiquetas que necesite.
12. Elija Guardar para guardar las etiquetas que haya añadido.

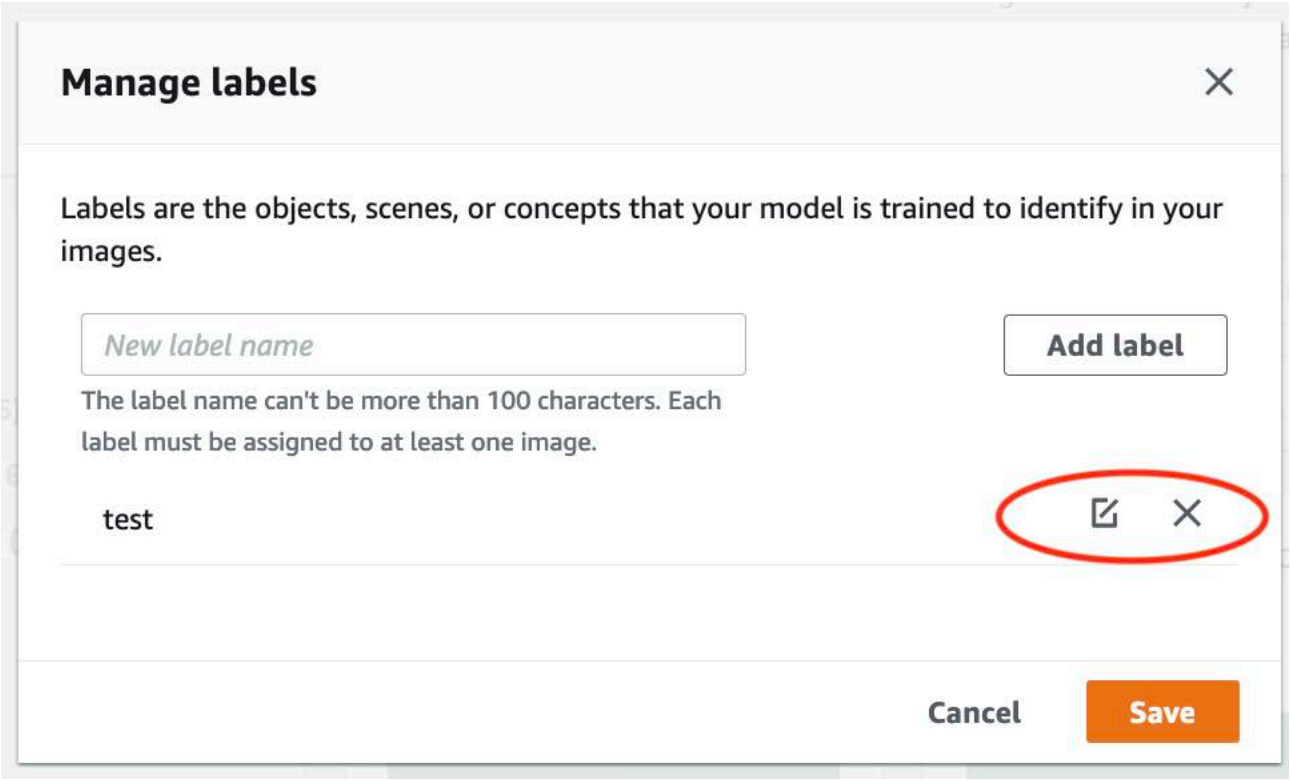
Cambiar y eliminar etiquetas (consola)

Puede cambiar el nombre de las etiquetas o eliminarlas después de agregarlas a un conjunto de datos. Solo puede eliminar las etiquetas que no estén asignadas a ninguna imagen.

Cómo cambiar el nombre de una etiqueta existente o eliminarla (consola)

1. Abra la consola Amazon Rekognition en. https://console.aws.amazon.com/rekognition/
2. Elija Usar etiquetas personalizadas.
3. Elija Comenzar.
4. En el panel de navegación izquierdo, elija Proyectos.

5. En la página Proyectos, elija el proyecto que desee usar. Se abrirá la página de detalles del proyecto.
6. Si quiere cambiar o eliminar etiquetas en el conjunto de datos de entrenamiento, elija la pestaña Entrenamiento. De lo contrario, seleccione la pestaña Prueba para cambiar o eliminar etiquetas en el conjunto de datos de prueba.
7. Elija Comenzar a etiquetar para activar el modo de etiquetado.
8. En la sección Etiquetas de la galería de conjuntos de datos, elija Administrar etiquetas para abrir el cuadro de diálogo Administrar etiquetas.
9. Elija la etiqueta que desee editar o eliminar.



   a. Si selecciona el icono de eliminación (X), la etiqueta se quitará de la lista.
   b. Si quiere cambiar la etiqueta, seleccione el icono de edición (lápiz y bloc de notas) e introduzca un nombre nuevo para la etiqueta en el cuadro de edición.
10. Elija Guardar para guardar los cambios.

Administración de etiquetas (SDK)

No hay una API única que administre las etiquetas de los conjuntos de datos. Si crea un conjunto de datos con `CreateDataset`, las etiquetas que se encuentren en el archivo de manifiesto o el

conjunto de datos copiado, se creará el grupo inicial de etiquetas. Si agrega más imágenes con la API de `UpdateDatasetEntries`, las nuevas etiquetas que se encuentren en las entradas se agregarán al conjunto de datos. Para obtener más información, consulte Agregar más imágenes (SDK). Para eliminar etiquetas de un conjunto de datos, es necesario eliminar todas las anotaciones de etiquetas del conjunto de datos.

Cómo eliminar etiquetas de un conjunto de datos

1. Llame a `ListDatasetEntries` para obtener las entradas del conjunto de datos. Para ver el código de ejemplo, consulte Listado de entradas del conjunto de datos (SDK).
2. En el archivo, elimine cualquier anotación en la etiqueta. Para obtener más información, consulte Importación de etiquetas de imagen en los archivos de manifiesto y the section called "Localización de objetos en archivos de manifiesto".
3. Use el archivo para actualizar el conjunto de datos con la API de `UpdateDatasetEntries`. Para obtener más información, consulte Agregar más imágenes (SDK).

Asignación de etiquetas de imagen a una imagen

Las etiquetas de imagen se utilizan para entrenar modelos que clasifican las imágenes en categorías. Una etiqueta de imagen indica que una imagen contiene un objeto, una escena o un concepto. Por ejemplo, en la siguiente imagen se ve un río. Si el modelo clasifica las imágenes como imágenes que contienen ríos, se debe agregar la etiqueta de imagen río. Para obtener más información, consulte Finalidad de los conjuntos de datos.

Un conjunto de datos que contiene etiquetas de imagen necesita tener definidas al menos dos etiquetas. Cada imagen necesita al menos una etiqueta asignada que identifique el objeto, la escena o el concepto de la imagen.

Cómo asignar etiquetas de imagen a una imagen (consola)

1. Abra la consola Amazon Rekognition en. https://console.aws.amazon.com/rekognition/
2. Elija Usar etiquetas personalizadas.
3. Elija Comenzar.
4. En el panel de navegación izquierdo, elija Proyectos.
5. En la página Proyectos, elija el proyecto que desee usar. Se abrirá la página de detalles del proyecto.
6. En el panel de navegación de la izquierda, elija Conjunto de datos.

7. Si quiere agregar etiquetas al conjunto de datos de entrenamiento, elija la pestaña Entrenamiento. Si no, seleccione la pestaña Prueba para agregar etiquetas al conjunto de datos de prueba.
8. Elija Comenzar a etiquetar para activar el modo de etiquetado.
9. En la galería de imágenes, seleccione una o varias imágenes a las que quiera agregar etiquetas. Solo puede seleccionar imágenes en una sola página a la vez. Cómo seleccionar una serie contigua de imágenes en una página:
    a. Seleccione la primera imagen del rango.
    b. Deje pulsada la tecla Mayús.
    c. Seleccione la última imagen del rango. Se seleccionarán también las imágenes entre la primera y la segunda imagen.
    d. Suelte la tecla Mayús.
10. Elija Asignar etiquetas de imagen.
11. En el cuadro de diálogo Asignar etiqueta de imagen a las imágenes seleccionadas, seleccione la etiqueta que desee asignar a la imagen o imágenes.
12. Elija Asignar para asignar la etiqueta a la imagen.
13. Repita el proceso de etiquetado hasta que todas las imágenes estén anotadas con las etiquetas necesarias.
14. Elija Guardar cambios para guardar los cambios.

Asignación de etiquetas de imagen (SDK)

Puede usar la API de `UpdateDatasetEntries` para agregar o actualizar las etiquetas de imagen que estén asignadas a una imagen. `UpdateDatasetEntries` toma una o varias líneas JSON. Cada línea JSON representa una sola imagen. En el caso de una imagen con una etiqueta de imagen, la línea JSON tiene un aspecto similar al siguiente.

```
{"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"}}
```

El campo `source-ref` indica la ubicación de la imagen. La línea JSON también incluye las etiquetas imagen asignadas a la imagen. Para obtener más información, consulte the section called "Importación de etiquetas de imagen en los archivos de manifiesto".

Cómo asignar etiquetas de imagen a una imagen

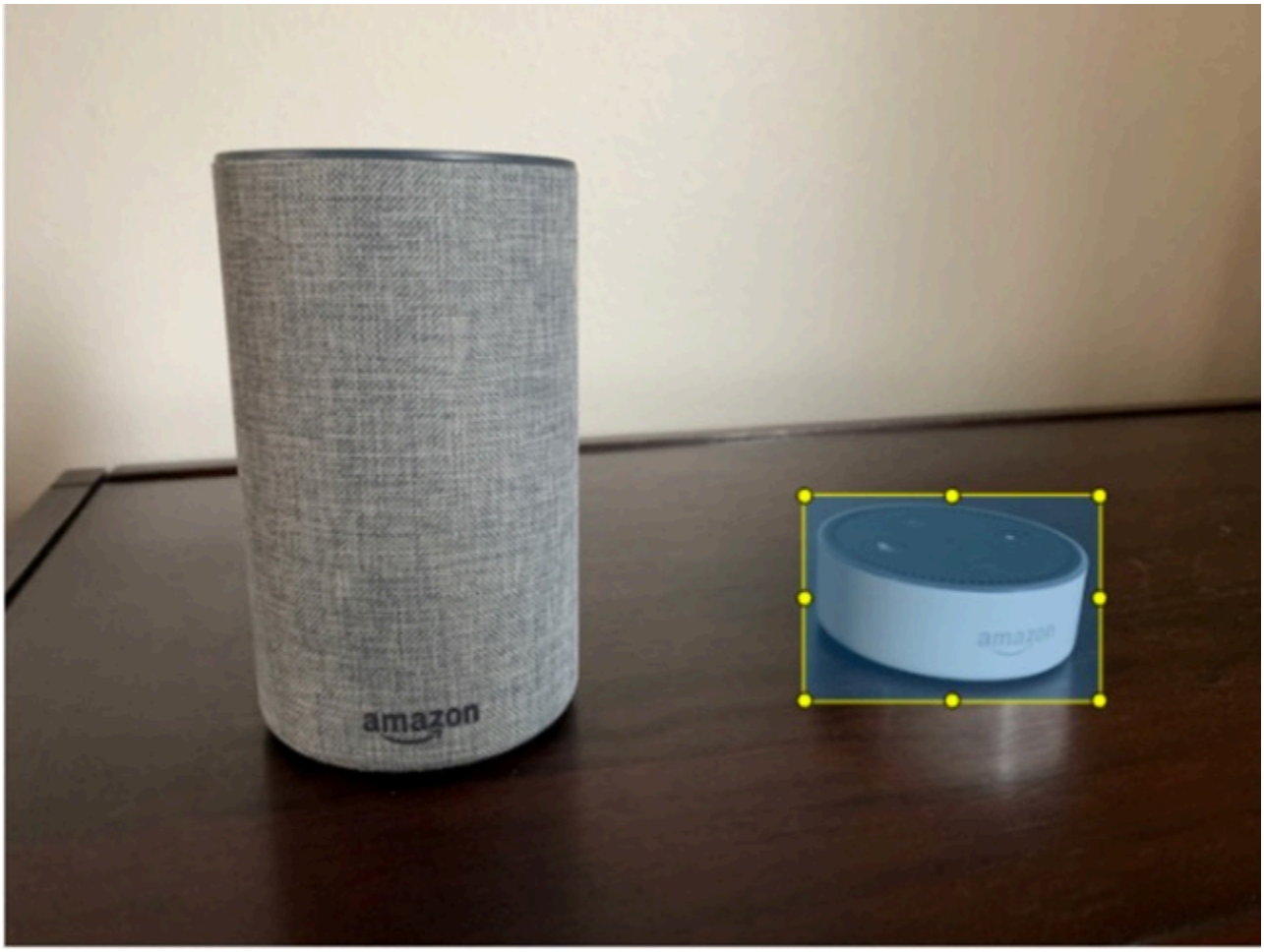
1. Obtenga la línea get JSON para la imagen existente con `ListDatasetEntries`. En el campo `source-ref`, indique la ubicación de la imagen a la que desea asignar la etiqueta. Para obtener más información, consulte Listado de entradas del conjunto de datos (SDK).
2. Actualice la línea JSON devuelta en el paso anterior con la información que se encuentra en Importación de etiquetas de imagen en los archivos de manifiesto.
3. Llame a `UpdateDatasetEntries` para actualizar la imagen. Para obtener más información, consulte Agregar más imágenes a un conjunto de datos.

Etiquetado de objetos con cuadros delimitadores

Si desea que el modelo detecte la ubicación de objetos dentro de una imagen, debe identificar qué es el objeto y en qué parte de la imagen se encuentra. Un cuadro delimitador es un cuadro que aísla un objeto de una imagen. Los cuadros delimitadores se utilizan para entrenar un modelo para que detecte distintos objetos en la misma imagen. El objeto se identifica asignando una etiqueta al cuadro delimitador.

> **Note**
>
> Si está entrenando un modelo para que busque objetos, escenas y conceptos con etiquetas de imagen, no es necesario realizar este paso.

Por ejemplo, si quiere entrenar un modelo que detecte dispositivos Amazon Echo Dot, dibuje un recuadro delimitador alrededor de cada Echo Dot en una imagen y asigna una etiqueta denominada Echo Dot al cuadro delimitador. En la siguiente imagen se muestra un cuadro delimitador alrededor de un Amazon Echo Dot. En la imagen también aparece un Amazon Echo sin un cuadro delimitador.



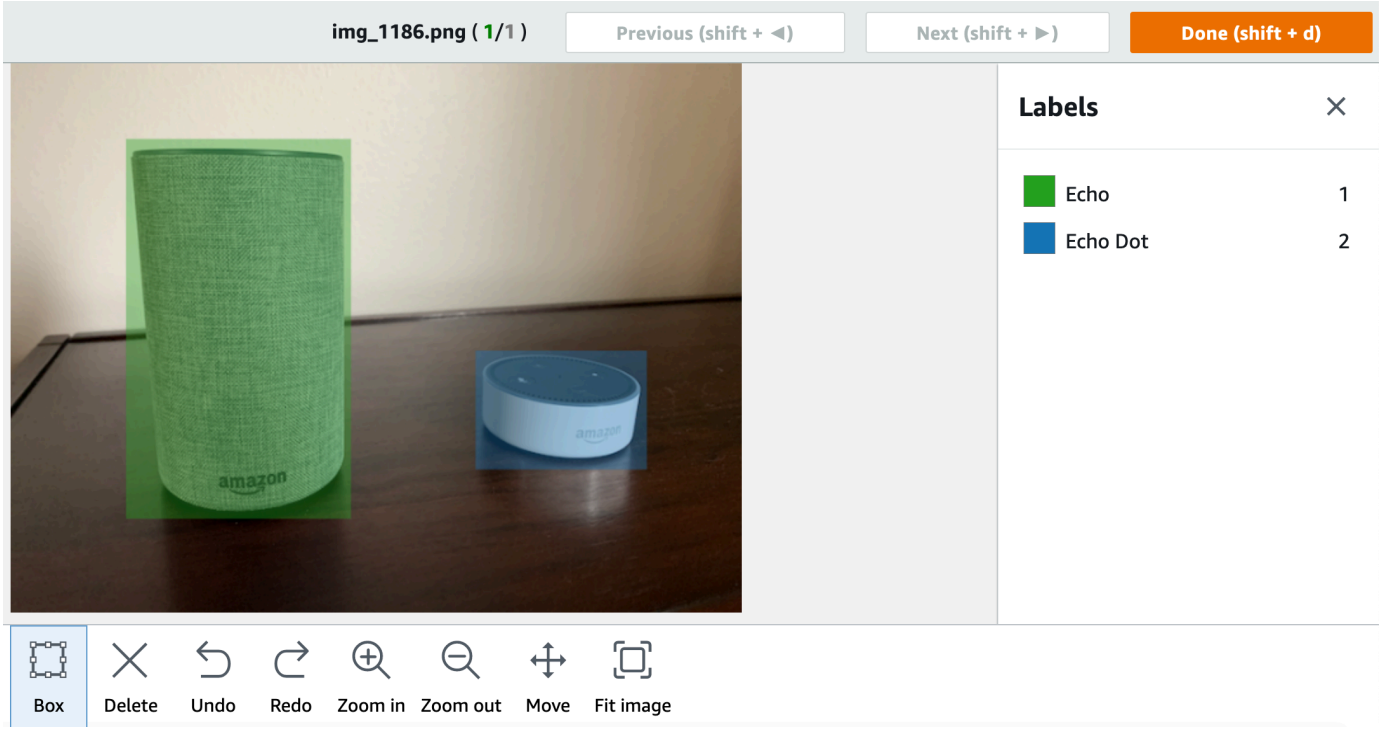
Localización de objetos con cuadros delimitadores (consola)

En este procedimiento, utilizará la consola para dibujar cuadros delimitadores alrededor de los objetos de las imágenes. Puede identificar objetos dentro de la imagen asignando etiquetas al cuadro delimitador.

> **Note**
>
> No es posible usar el navegador Safari para agregar cuadros delimitadores a las imágenes. Para ver los navegadores compatibles, consulte Configuración de Etiquetas personalizadas de Amazon Rekognition.

Para poder agregar cuadros delimitadores, debe agregar al menos una etiqueta al conjunto de datos. Para obtener más información, consulte Agregar etiquetas nuevas (consola).

1. Abra la consola Amazon Rekognition en. https://console.aws.amazon.com/rekognition/
2. Elija Usar etiquetas personalizadas.
3. Elija Comenzar.
4. En el panel de navegación izquierdo, elija Proyectos.
5. En la página Proyectos, elija el proyecto que desee usar. Se abrirá la página de detalles del proyecto.
6. En la página de detalles del proyecto, elija Etiquetar imágenes.
7. Si quiere agregar cuadros delimitadores a las imágenes del conjunto de datos de entrenamiento, elija la pestaña Entrenamiento. Si no, seleccione la pestaña Prueba para agregar cuadros delimitadores a las imágenes del conjunto de datos de prueba.
8. Elija Comenzar a etiquetar para activar el modo de etiquetado.
9. En la galería de imágenes, elija a las que desee agregar cuadros delimitadores.
10. Seleccione Dibujar cuadro delimitador. Se verán una serie de consejos antes de que aparezca el editor de cuadros delimitadores.
11. En el panel Etiquetas de la derecha, seleccione la etiqueta que desee asignar a un cuadro delimitador.
12. En la herramienta de dibujo, coloque el puntero en el área superior izquierda del objeto deseado.
13. Pulse el botón izquierdo del ratón y dibuje un cuadro alrededor del objeto. Dibuje el cuadro delimitador lo más cerca posible del objeto.
14. Suelte el botón del ratón. El cuadro delimitador aparecerá resaltado.
15. Elija Siguiente si tiene más imágenes que etiquetar. Si ya ha terminado, seleccione Listo para terminar de etiquetar.



16. Repita los pasos del 1 al 7 hasta que haya creado un cuadro delimitador en cada imagen que contenga objetos.
17. Elija Guardar cambios para guardar los cambios.
18. Seleccione Salir para salir del modo de etiquetado.

Localización de objetos con cuadros delimitadores (SDK)

Puede usar la API de `UpdateDatasetEntries` para agregar o actualizar la información de la ubicación de objetos de una imagen. `UpdateDatasetEntries` toma una o varias líneas JSON. Cada línea JSON representa una sola imagen. En la localización de objetos, una línea JSON tiene un aspecto similar al siguiente.

```
{"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"}}
```

El campo `source-ref` indica la ubicación de la imagen. La línea JSON también incluye cuadros delimitadores etiquetados por cada objeto de la imagen. Para obtener más información, consulte the section called “Localización de objetos en archivos de manifiesto”.

Cómo asignar cuadros delimitadores a una imagen

1. Obtenga la línea get JSON para la imagen existente con `ListDatasetEntries`. En el campo `source-ref`, indique la ubicación de la imagen a la que desea asignar la etiqueta de imagen. Para obtener más información, consulte Listado de entradas del conjunto de datos (SDK).
2. Actualice la línea JSON devuelta en el paso anterior con la información que se encuentra en Localización de objetos en archivos de manifiesto.
3. Llame a `UpdateDatasetEntries` para actualizar la imagen. Para obtener más información, consulte Agregar más imágenes a un conjunto de datos.

Depuración de errores de conjuntos de datos

Durante la creación del conjunto de datos, se pueden producir dos tipos de errores: errores terminales y errores no terminales. Los errores terminales pueden detener la creación o actualización del conjunto de datos. Los errores no terminales no detienen la creación o actualización del conjunto de datos.

Temas

- Depuración de errores terminales de conjunto de datos
- Depuración de errores no terminales de conjunto de datos

Depuración de errores terminales de conjunto de datos

Existen dos tipos de errores terminales: errores de archivo que provocan un error en la creación del conjunto de datos y errores de contenido que Etiquetas personalizadas de Amazon Rekognition elimina del conjunto de datos. No se pueden crear conjuntos de datos si hay demasiados errores de contenido.

Temas

- Errores de archivo terminales
- Errores terminales de contenido

Errores de archivo terminales

Los siguientes son errores de archivo. Puede obtener información sobre los errores de archivos llamando a `DescribeDataset` y comprobando los campos `Status` y `StatusMessage`. Para ver el código de ejemplo, consulte Descripción de un conjunto de datos (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_INSUFICIENT_IMAGES_PER_LABEL_FOR_DISTRIBUTE

ERROR_MANIFEST_INACCESSIBLE_OR_UNSUPPORTED_FORMAT

Mensaje de error

The manifest file extension or contents are invalid.

El archivo de manifiesto de entrenamiento o de prueba no tiene una extensión de archivo o su contenido no es válido.

Cómo corregir el error ERROR_MANIFEST_INACCESSIBLE_OR_UNSUPPORTED_FORMAT

- Compruebe las siguientes causas posibles en los archivos de manifiesto de entrenamiento y de prueba.
  - El archivo de manifiesto no tiene la extensión de archivo. Por convención, la extensión del archivo es `.manifest`.
  - No se ha encontrado el bucket de Amazon S3 o la clave del archivo de manifiesto.

ERROR_MANIFEST_SIZE_TOO_LARGE

Mensaje de error

The manifest file size exceeds the maximum supported size.

El tamaño del archivo de manifiesto de entrenamiento o de prueba (en bytes) es demasiado grande. Para obtener más información, consulte Directrices y cuotas en Etiquetas personalizadas de Amazon Rekognition. Un archivo de manifiesto puede tener menos de la cantidad máxima de líneas JSON y aun así superar el tamaño máximo de archivo.

No puede usar la consola de Etiquetas personalizadas de Amazon Rekognition para corregir el error El tamaño del archivo de manifiesto supera el tamaño máximo admitido.

Cómo corregir el error ERROR_MANIFEST_SIZE_TOO_LARGE

1. Compruebe cuáles de los manifiestos de entrenamiento y de prueba superan el tamaño máximo de archivo.
2. Reduzca el número de líneas JSON demasiado grandes en los archivos de manifiesto. Para obtener más información, consulte Creación de un archivo de manifiesto.

ERROR_MANIFEST_ROWS_EXCEEDS_MAXIMUM

Mensaje de error

The manifest file has too many rows.

Más información

El número de líneas JSON (número de imágenes) del archivo de manifiesto supera el límite permitido. El límite es diferente en los modelos de imágenes y los modelos de ubicación de objetos. Para obtener más información, consulte Directrices y cuotas en Etiquetas personalizadas de Amazon Rekognition.

Los errores en las líneas JSON se validan hasta que el número de líneas JSON alcanza el límite de `ERROR_MANIFEST_ROWS_EXCEEDS_MAXIMUM`.

No es posible usar la consola de Etiquetas personalizadas de Amazon Rekognition para subsanar el error `ERROR_MANIFEST_ROWS_EXCEEDS_MAXIMUM`.

Cómo corregir **`ERROR_MANIFEST_ROWS_EXCEEDS_MAXIMUM`**

- Reduzca el número de líneas JSON en el manifiesto. Para obtener más información, consulte Creación de un archivo de manifiesto.

ERROR_INVALID_PERMISSIONS_MANIFEST_S3_BUCKET

Mensaje de error

The S3 bucket permissions are incorrect.

Etiquetas personalizadas de Amazon Rekognition no tiene permisos para uno o varios de los buckets que contienen los archivos de manifiesto de entrenamiento y de prueba.

No es posible usar la consola de Etiquetas personalizadas de Amazon Rekognition para subsanar este error.

Cómo corregir el error ERROR_INVALID_PERMISSIONS_MANIFEST_S3_BUCKET

- Compruebe los permisos de los buckets que contengan los manifiestos de entrenamiento y de prueba. Para obtener más información, consulte Paso 2: Configurar los permisos de la consola de Etiquetas personalizadas de Amazon Rekognition.

ERROR_TOO_MANY_RECORDS_IN_ERROR

Mensaje de error

The manifest file has too many terminal errors.

Cómo corregir **ERROR_TOO_MANY_RECORDS_IN_ERROR**

- Reduzca el número de líneas JSON (imágenes) con errores terminales en el contenido. Para obtener más información, consulte Errores terminales de contenido del manifiesto.

No es posible usar la consola de Etiquetas personalizadas de Amazon Rekognition para subsanar este error.

ERROR_MANIFEST_TOO_MANY_LABELS

Mensaje de error

The manifest file has too many labels.

Más información

El número de etiquetas únicas en el manifiesto (conjunto de datos) supera el límite permitido. Si el conjunto de datos de entrenamiento se divide para crear un conjunto de datos de prueba, el número de etiquetas se determinará después de la división.

Cómo corregir ERROR_MANIFEST_TOO_MANY_LABELS (consola)

- Elimine las etiquetas del conjunto de datos. Para obtener más información, consulte Administración de etiquetas. Las etiquetas se eliminan automáticamente de las imágenes y los cuadros delimitadores en el conjunto de datos.

Cómo corregir ERROR_MANIFEST_TOO_MANY_LABELS (línea JSON)

- Manifiestos con líneas JSON de imagen: si la imagen tiene una sola etiqueta, elimine las líneas JSON de las imágenes que utilicen la etiqueta deseada. Si la línea JSON contiene varias etiquetas, elimine solo el objeto JSON de la etiqueta deseada. Para obtener más información, consulte Cómo agregar varias etiquetas de imagen a una imagen.

  Manifiestos con líneas JSON con la ubicación del objeto: elimine el cuadro delimitador y la información de etiqueta asociada a la etiqueta que desee eliminar. Haga esto en cada línea JSON que contenga la etiqueta deseada. Es necesario eliminar la etiqueta de la matriz `class-map` y los objetos correspondientes de la matriz `objects` y `annotations`. Para obtener más información, consulte Localización de objetos en archivos de manifiesto.

ERROR_INSUFICIENT_IMAGES_PER_LABEL_FOR_DISTRIBUTE

Mensaje de error

The manifest file doesn't have enough labeled images to distribute the dataset.

La distribución del conjunto de datos se produce cuando Etiquetas personalizadas de Amazon Rekognition divide un conjunto de datos de entrenamiento para crear un conjunto de datos de prueba. También puede dividir un conjunto de datos llamando a la API de `DistributeDatasetEntries`.

Cómo corregir el error ERROR_MANIFEST_TOO_MANY_LABELS

- Agregar más imágenes etiquetadas al conjunto de datos de entrenamiento

Errores terminales de contenido

Los siguientes son errores terminales de contenido. Al crear el conjunto de datos, las imágenes que tienen errores terminales de contenido se eliminan del conjunto de datos. El conjunto de datos se puede seguir utilizando para el entrenamiento. Si hay demasiados errores de contenido, el conjunto de datos o los cambios no funcionarán. Los errores terminales de contenido relacionados con las operaciones de los conjuntos de datos no aparecen en la consola ni se devuelven de `DescribeDataset` ni ninguna otra API. Si faltan imágenes o anotaciones en los conjuntos de datos, consulte los archivos de manifiesto de los conjuntos de datos para ver si se dan los siguientes problemas:

- La longitud de una línea JSON es demasiado larga. La longitud máxima es de 100 000 caracteres.
- Falta el valor `source-ref` en una línea JSON.
- El formato del valor `source-ref` en una línea JSON no es válido.
- El contenido de una línea JSON no es válido.
- Hay un valor en el que el campo `source-ref` aparece más de una vez. Solo se puede hacer referencia a una imagen una vez en un conjunto de datos.

Para obtener más información sobre el campo `source-ref`, consulte Creación de un archivo de manifiesto.

Depuración de errores no terminales de conjunto de datos

Los siguientes son errores no terminales que pueden producirse durante la creación o actualización del conjunto de datos. Estos errores pueden invalidar una línea JSON completa o invalidar las anotaciones dentro de una línea JSON. Si una línea JSON presenta un error, no se usará para el entrenamiento. Si una anotación dentro de una línea JSON contiene un error, la línea JSON se seguirá utilizando para el entrenamiento, pero sin la anotación rota. Para obtener más información sobre la líneas JSON, consulte Creación de un archivo de manifiesto.

Puede acceder a los errores no terminales relacionados en la consola y llamando a la API de `ListDatasetEntries`. Para obtener más información, consulte Listado de entradas del conjunto de datos (SDK).

Los siguientes errores también se devuelven durante el entrenamiento. Le recomendamos que subsane estos errores antes de entrenar el modelo. Para obtener más información, consulte. Errores no terminales de validación en líneas JSON

- 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

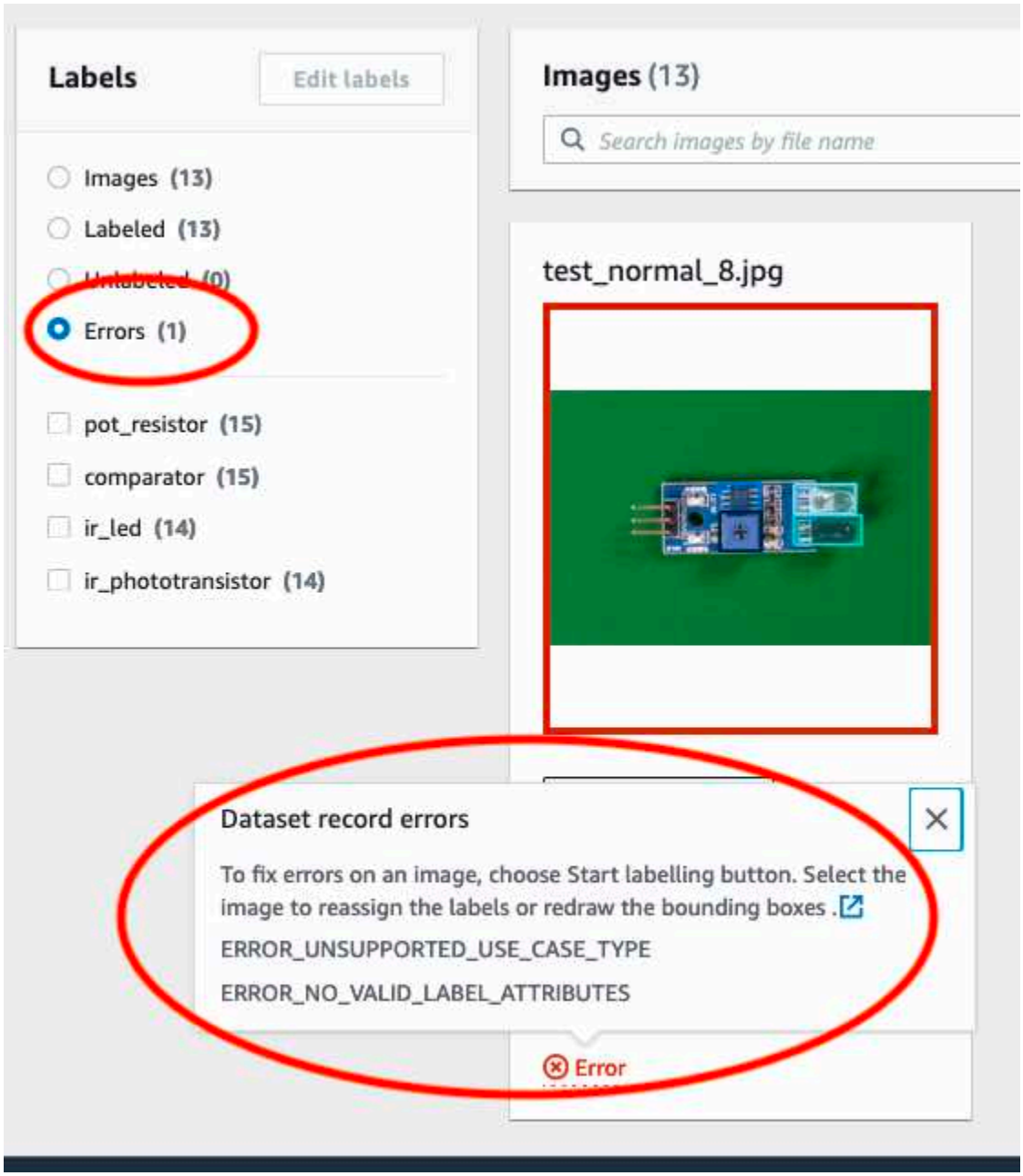
Acceso a los errores no terminales

Puede usar la consola para saber qué imágenes de un conjunto de datos tienen errores no terminales. También puede llamar a la API de `ListDatasetEntries` para ver los mensajes de error. Para obtener más información, consulte Listado de entradas del conjunto de datos (SDK).

Cómo acceder a los errores no terminales (consola)

1. Abra la consola Amazon Rekognition en. https://console.aws.amazon.com/rekognition/
2. Elija Usar etiquetas personalizadas.
3. Elija Comenzar.
4. En el panel de navegación izquierdo, elija Proyectos.
5. En la página Proyectos, elija el proyecto que desee usar. Se abrirá la página de detalles del proyecto.

6. Si quiere ver los errores no terminales en el conjunto de datos de entrenamiento, seleccione la pestaña Entrenamiento. De lo contrario, elija la pestaña Prueba para ver los errores no terminales en el conjunto de datos de prueba.
7. En la sección Etiquetas de la galería de conjuntos de datos, elija Errores. La galería de conjuntos de datos se filtrará para que salgan solamente las imágenes con errores.
8. Seleccione Error debajo de cada imagen para ver el código de error. Use la información de Errores no terminales de validación en líneas JSON para corregir el error.



Entrenamiento de un modelo de Etiquetas personalizadas de Amazon Rekognition

Puede entrenar un modelo a través de la consola de Etiquetas personalizadas de Amazon Rekognition o mediante la API de Etiquetas personalizadas de Amazon Rekognition. Si el entrenamiento del modelo no funciona, utilice la información incluida en Depuración de un modelo de entrenamiento con errores para hallar la causa del error.

> **Note**
>
> Se le cobrará por el tiempo que se tarde en entrenar correctamente un modelo. Por lo general, el entrenamiento tarda entre 30 minutos y 24 horas en completarse. Para obtener más información, consulte Horas de entrenamiento.

Se crea una nueva versión del modelo cada vez que se entrena. Etiquetas personalizadas de Amazon Rekognition crea un nombre para el modelo que es una combinación del nombre del proyecto y la marca de tiempo de creación del modelo.

Para entrenar el modelo, Etiquetas personalizadas de Amazon Rekognition hace una copia de las imágenes de entrenamiento y de prueba de origen. De forma predeterminada, las imágenes copiadas se cifran en reposo con una clave que AWS posee y administra. También puede optar por utilizar su propia AWS KMS key. Si usa su propia clave de KMS, necesitará los siguientes permisos en la clave de KMS.

- km: CreateGrant
- km: DescribeKey

Para obtener más información, consulte Conceptos de AWS Key Management Service. Las imágenes de origen no se ven afectadas.

Puede utilizar el cifrado del servidor de KMS (SSE-KMS) para cifrar las imágenes de entrenamiento y de prueba del bucket de Amazon S3 antes de que las copie Etiquetas personalizadas de Amazon Rekognition. Para permitir que Amazon Rekognition Custom Labels acceda a sus imágenes AWS , su cuenta necesita los siguientes permisos en la clave de KMS.

- kms: GenerateDataKey

- kms:Decrypt

Para obtener más información, consulte Protección de datos mediante el cifrado del servidor con claves de KMS almacenadas en AWS Key Management Service (SSE-KMS).

Tras entrenar un modelo, puede evaluar su rendimiento y realizar mejoras. Para obtener más información, consulte Mejora de un modelo de Etiquetas personalizadas de Amazon Rekognition.

Para realizar otras tareas del modelo, como etiquetar un modelo, consulte Administración de un modelo de Etiquetas personalizadas de Amazon Rekognition.

Temas

Entrenamiento de un modelo (consola)

Puede usar la consola de Etiquetas personalizadas de Amazon Rekognition para entrenar un modelo.

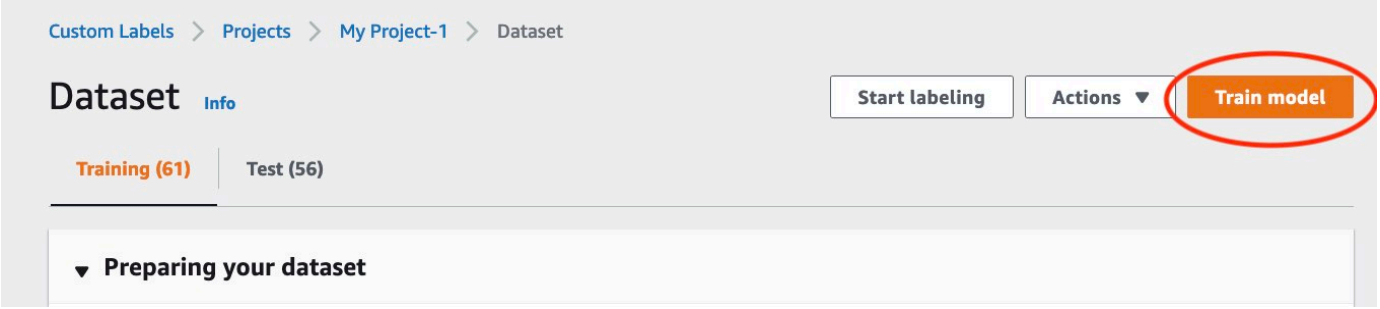
El entrenamiento requiere un proyecto con un conjunto de datos de entrenamiento y un conjunto de datos de prueba. Si el proyecto no tiene un conjunto de datos de prueba, la consola de Etiquetas personalizadas de Amazon Rekognition dividirá el conjunto de datos de entrenamiento durante el entrenamiento para crear uno para su proyecto. Las imágenes elegidas son una muestra representativa y no se utilizan en el conjunto de datos de entrenamiento. Le recomendamos que divida su conjunto de datos de entrenamiento solo si no tiene un conjunto de datos de prueba alternativo que pueda usar. Al dividir un conjunto de datos de entrenamiento, se reduce la cantidad de imágenes disponibles para el entrenamiento.

> **Note**
>
> Se le cobrará por el tiempo que se tarde en entrenar un modelo. Para obtener más información, consulte Horas de entrenamiento.

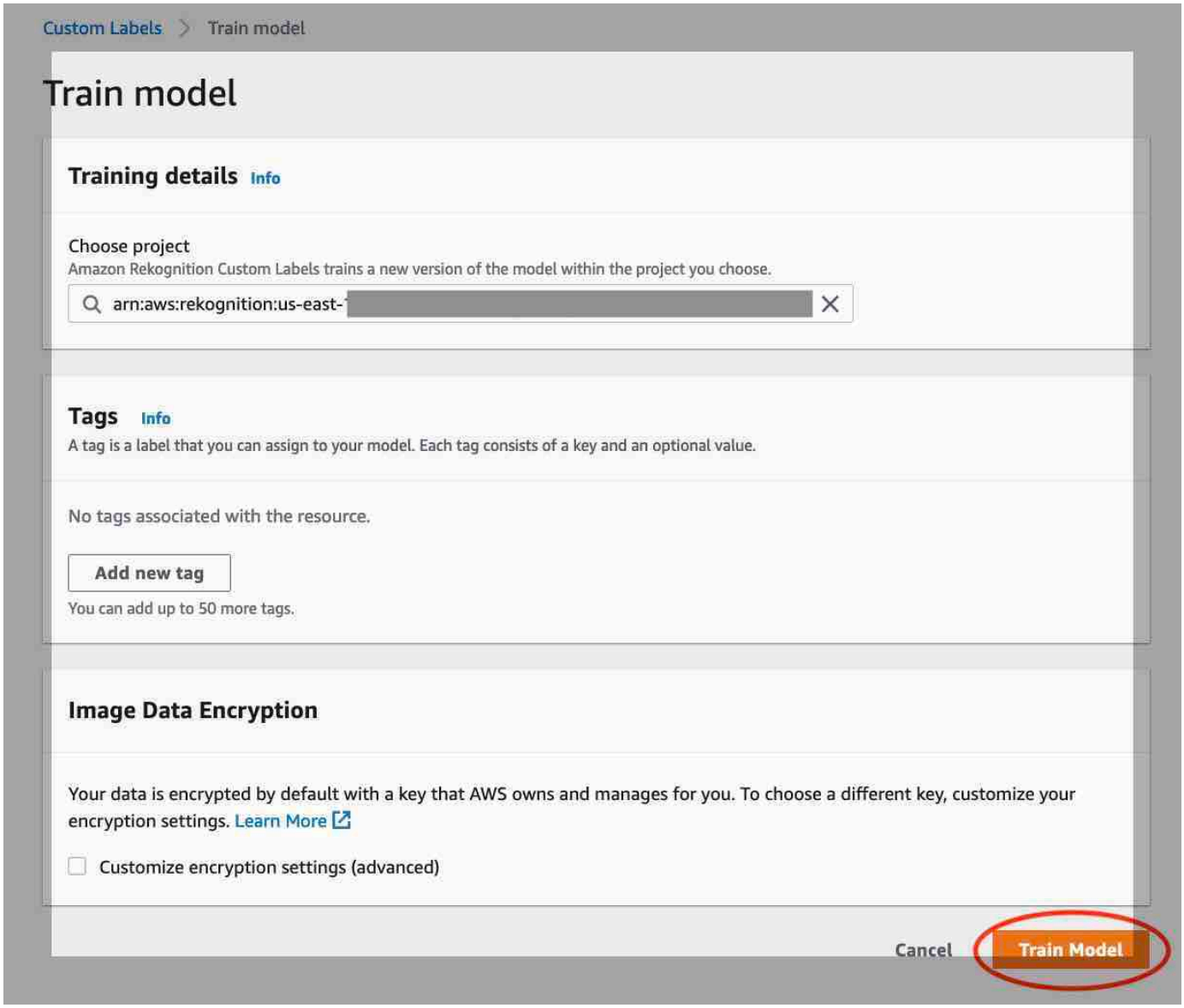
Para entrenar su modelo (consola)

1. Abra la consola Amazon Rekognition en. https://console.aws.amazon.com/rekognition/

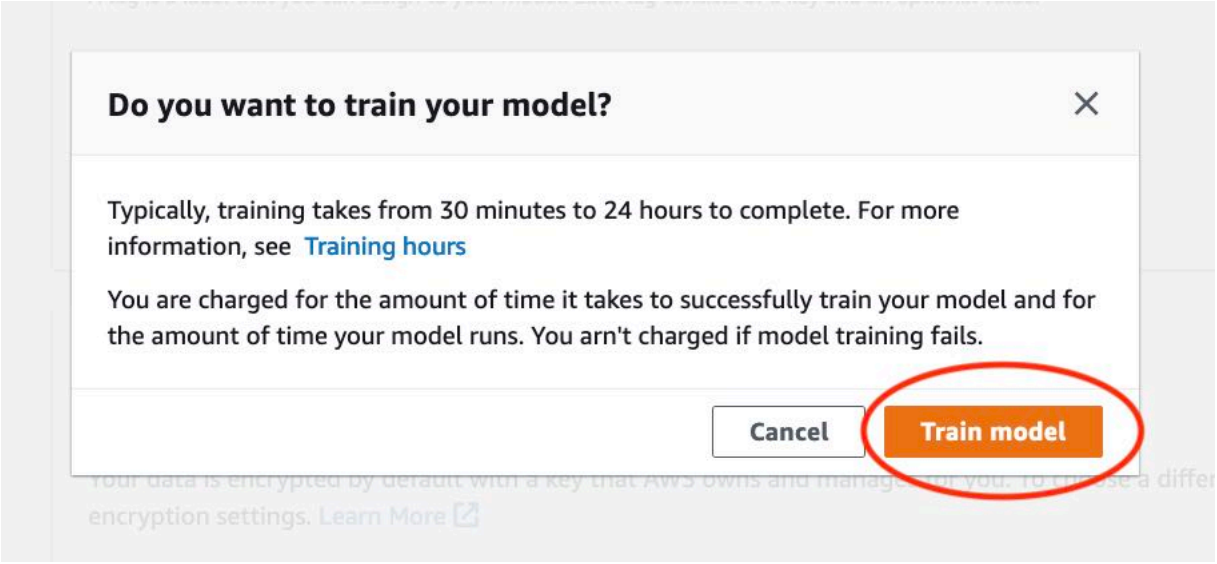
2. Elija Usar etiquetas personalizadas.
3. En el panel de navegación izquierdo, elija Proyectos.
4. En la página Proyectos, elija el proyecto que contiene el modelo entrenado que desee entrenar.
5. En la página Proyecto, elija Entrenar modelo.



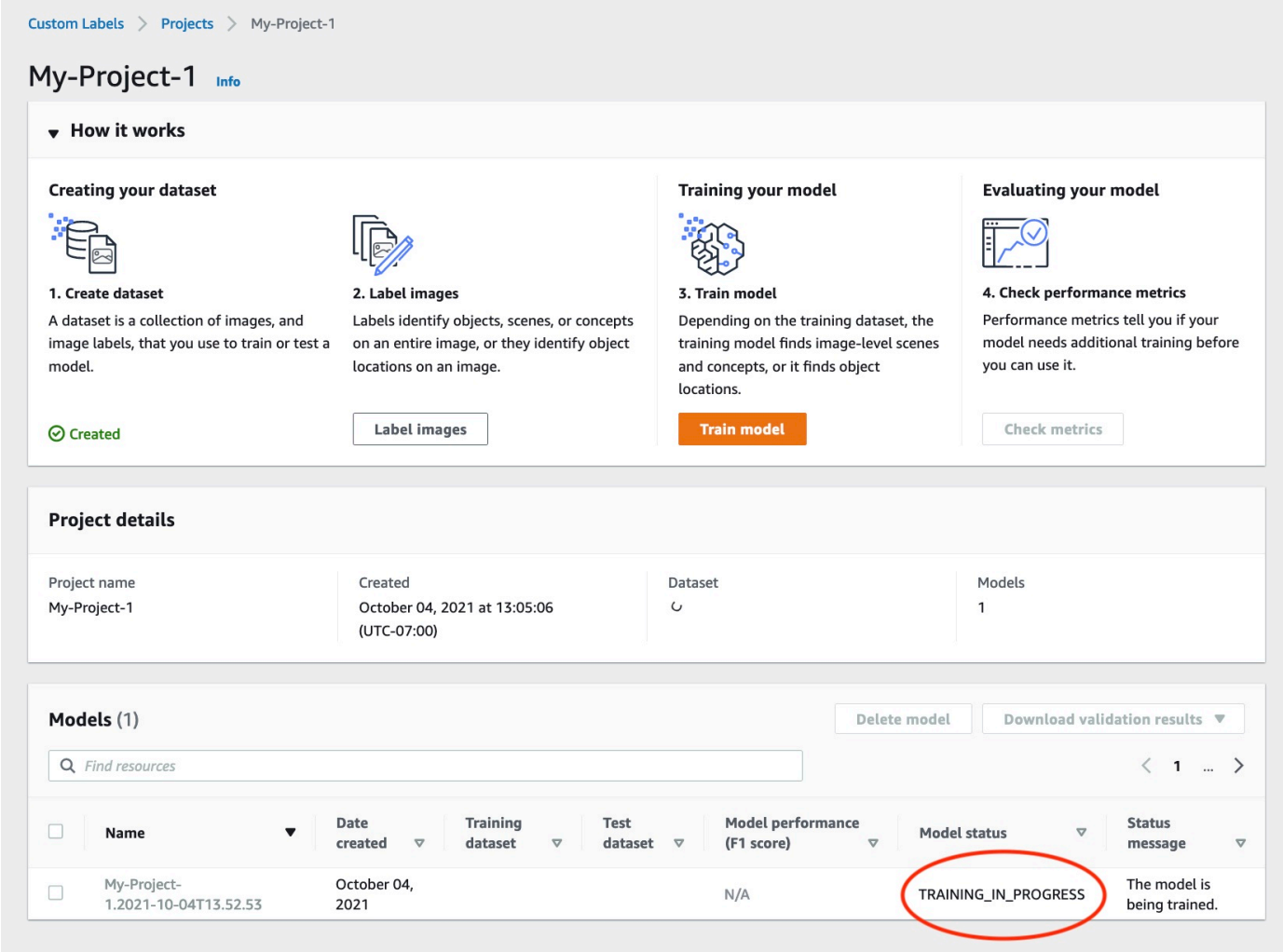
6. (Opcional) Si quiere usar su propia clave de cifrado de AWS KMS, haga lo siguiente:
   a. En Cifrado de datos de imagen, elija Personalizar la configuración de cifrado (avanzado).
   b. En encryption.aws_kms_key, introduzca el nombre de recurso de Amazon (ARN) de su clave o elija una clave de AWS KMS existente. Para crear una clave nueva, elija Crear una clave de AWS IMS.
7. (Opcional) Si desea agregar etiquetas al modelo, haga lo siguiente:
   a. En la sección Etiquetas, elija Agregar nueva etiqueta.
   b. Introduzca lo siguiente:
      i. El nombre de la clave en Key.
      ii. El valor de la clave en Valor.
   c. Para añadir más etiquetas, repita los pasos 6a y 6b.
   d. (Opcional) Si desea eliminar una etiqueta, elija Eliminar junto a la etiqueta que desea eliminar. Si va a eliminar una etiqueta guardada anteriormente, se eliminará al guardar los cambios.
8. En la página Entrenar modelo, elija Entrenar modelo. El nombre de recurso de Amazon (ARN) del proyecto se encuentra en el cuadro de edición Elegir proyecto. Si no es así, introduzca el ARN del proyecto.



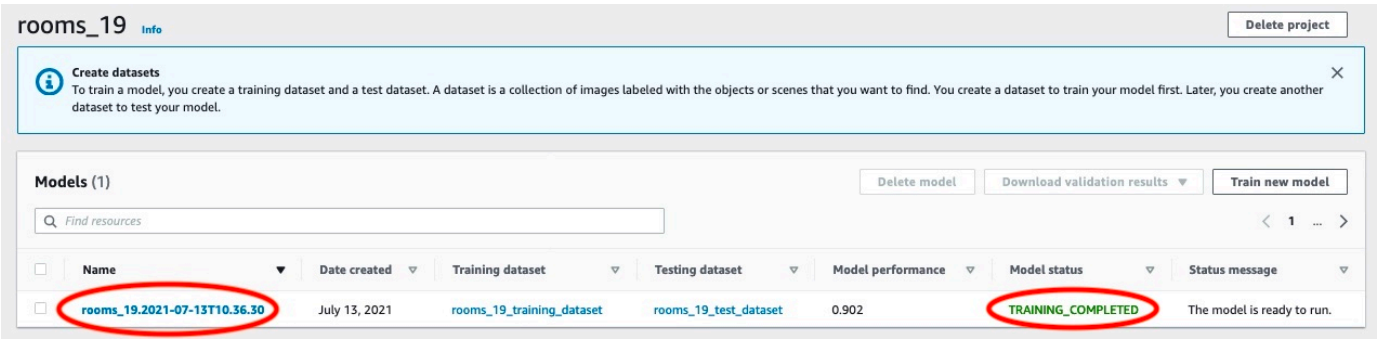
9. En el cuadro de diálogo ¿Quiere entrenar su modelo?, escoja Entrenar modelo.



10. En la sección Modelos de la página del proyecto, podrá comprobar el estado actual en la columna `Model Status` en la que se esté efectuando el entrenamiento. El entrenamiento de un modelo tarda un tiempo en completarse.



11. Una vez finalizado, elija el nombre del modelo. El entrenamiento termina cuando el estado del modelo es TRAINING_COMPLETED. Si hay algún problema con el entrenamiento, consulte [Depuración de un modelo de entrenamiento con errores](#).



12. Siguiente paso: Evaluar el modelo. Para obtener más información, [Mejora de un modelo de Etiquetas personalizadas de Amazon Rekognition](#).

Entrenamiento de un modelo (SDK)

Entrenas a un modelo llamando. CreateProjectVersion Para entrenar un modelo, se necesita la siguiente información:

- Nombre: un nombre único para la versión del modelo.
- ARN del proyecto: nombre de recurso de Amazon (ARN) del proyecto que administra el modelo.
- Ubicación de los resultados del entrenamiento: ubicación de Amazon S3 en la que se registran los resultados. Puede usar la misma ubicación que el bucket de Amazon S3 de la consola o puede elegir una ubicación diferente. Le recomendamos que elija una ubicación diferente porque así podrá configurar permisos y evitar posibles conflictos de nombres con los resultados del entrenamiento al utilizar la consola de Etiquetas personalizadas de Amazon Rekognition.

El entrenamiento utiliza los conjuntos de datos de entrenamiento y de prueba asociados al proyecto. Para obtener más información, consulte Administración de conjuntos de datos.

> **Note**
>
> Si lo desea, puede indicar los archivos de manifiesto de los conjuntos de datos de entrenamiento y de prueba que sean externos a un proyecto. Si abre la consola después de entrenar un modelo con archivos de manifiesto externos, Etiquetas personalizadas de Amazon Rekognition crea los conjuntos de datos automáticamente utilizando el último grupo de archivos de manifiesto utilizado para el entrenamiento. Ya se no podrá entrenar la versión de un modelo del proyecto si se especifican archivos de manifiesto externos. Para obtener más información, consulte CreatePrjectVersion.

La respuesta de `CreateProjectVersion` es un ARN que se utiliza para identificar la versión del modelo en solicitudes posteriores. También puede usar el ARN para proteger la versión del modelo. Para obtener más información, consulte Protección de proyectos de Etiquetas personalizadas de Amazon Rekognition.

El entrenamiento de la versión de un modelo tarda un tiempo en completarse. En los ejemplos de Python y Java de este tema, se utilizan esperadores para el entrenamiento del modelo. Un esperador es un método de utilidad que sondean si se da un determinado estado. También puede conocer el estado actual del entrenamiento llamando a `DescribeProjectVersions`. El entrenamiento habrá

finalizado cuando el valor del campo `Status` es `TRAINING_COMPLETED`. Una vez terminado el entrenamiento, puede evaluar la calidad del modelo revisando los resultados de la evaluación.

Entrenamiento de un modelo (SDK)

En el siguiente ejemplo se indica cómo entrenar un modelo mediante los conjuntos de datos de entrenamiento y de prueba asociados a un proyecto.

Cómo entrenar un modelo (SDK)

1. Si aún no lo ha hecho, instale y configure el AWS CLI y el AWS SDKs. Para obtener más información, consulte Paso 4: Configure y AWS CLIAWS SDKs.
2. Utilice el siguiente código de ejemplo para describir un proyecto.

   AWS CLI

   En el siguiente ejemplo se ve cómo se crea un modelo. El conjunto de datos de entrenamiento se divide para crear el conjunto de datos de prueba. Sustituya lo siguiente:

   - `my_project_arn` por el nombre de recurso de Amazon (ARN) del proyecto.
   - `version_name` por el nombre único de la versión único que prefiera.
   - `output_bucket` por el nombre del bucket de Amazon S3 en el que Etiquetas personalizadas de Amazon Rekognition guarda los resultados del entrenamiento.
   - `output_folder` por el nombre de la carpeta donde se guardan los resultados del entrenamiento.
   - (parámetro opcional) `--kms-key-id` por el identificador de la clave principal de cliente de AWS Key Management Service.

```
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

En el siguiente ejemplo se ve cómo se crea un modelo. Indique los siguientes argumentos de línea de comandos:

- `project_arn`: el nombre de recurso de Amazon (ARN) del proyecto.
- `version_name`: el nombre único de la versión del modelo que elija.
- `output_bucket`: el nombre del bucket de Amazon S3 en el que Etiquetas personalizadas de Amazon Rekognition guarda los resultados del entrenamiento.
- `output_folder`: el nombre de la carpeta donde se guardan los resultados del entrenamiento.

Si lo desea, indique los siguientes parámetros de línea de comandos para adjuntar una etiqueta al modelo:

- `tag`: el nombre de la etiqueta de su elección que quiera asociar al modelo.
- `tag_value`: el valor de la etiqueta.

```
#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

En el siguiente ejemplo se ve cómo se entrena un modelo. Indique los siguientes argumentos de línea de comandos:

- `project_arn`: el nombre de recurso de Amazon (ARN) del proyecto.
- `version_name`: el nombre único de la versión del modelo que elija.
- `output_bucket`: el nombre del bucket de Amazon S3 en el que Etiquetas personalizadas de Amazon Rekognition guarda los resultados del entrenamiento.
- `output_folder`: el nombre de la carpeta donde se guardan los resultados del entrenamiento.

```
/*
   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. Si hay algún problema con el entrenamiento, consulte Depuración de un modelo de entrenamiento con errores.

Depuración de un modelo de entrenamiento con errores

Es posible que se produzcan errores durante el entrenamiento del modelo. Amazon Rekognition Custom Labels informa de los errores de entrenamiento en la consola y en la respuesta de. DescribeProjectVersions

Los errores son terminales (el entrenamiento no puede continuar) o no terminales (el entrenamiento puede continuar). En el caso de errores relacionados con el contenido de los conjuntos de datos de entrenamiento y de prueba, puede descargar los resultados de la validación (un resumen del manifiesto y manifiestos de validación del entrenamiento y de las pruebas). Utilice los códigos de error de los resultados de la validación para obtener más información en esta sección. En esta sección también se da información sobre los errores del archivo de manifiesto (errores terminales que se producen antes de que se valide el contenido del archivo de manifiesto).

> **Note**
>
> Un manifiesto es el archivo que se utiliza para almacenar el contenido de un conjunto de datos.

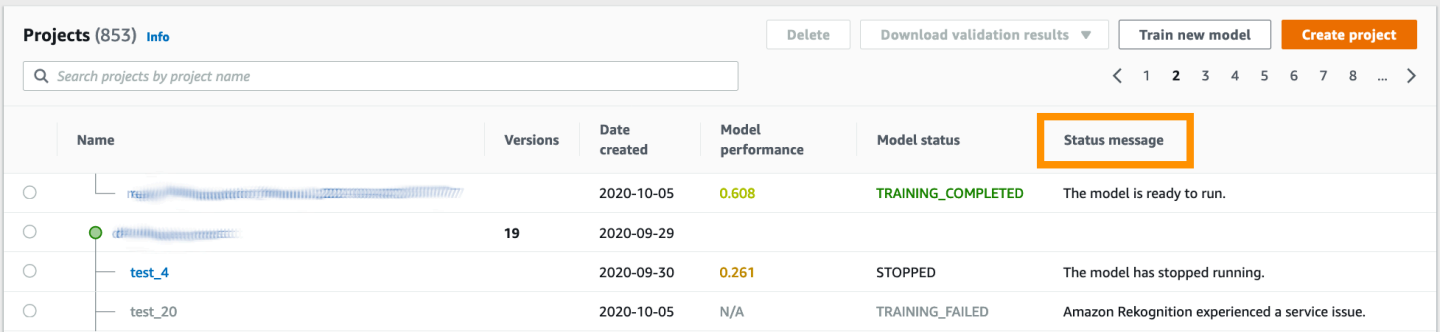
Puede subsanar algunos errores en la consola de Etiquetas personalizadas de Amazon Rekognition. Hay otros errores que necesitarán que se actualicen los archivos de manifiesto de entrenamiento o

de prueba. Otra vez, es posible que haya que realizar otros cambios, como en los permisos de IAM. Para obtener más información, consulte la documentación para ver cada uno de los errores.

Errores terminales

Los errores terminales hacen que se detenga el entrenamiento de un modelo. Existen tres categorías de errores de entrenamiento terminales: errores de servicio, errores de archivos de manifiesto y errores de contenido de manifiestos.

En la consola, Etiquetas personalizadas de Amazon Rekognition registra los errores terminales de un modelo en la columna Mensaje de estado de la página de proyectos. El panel de administración de proyectos muestra una lista de proyectos con el nombre, las versiones, la fecha de creación, el rendimiento del modelo y un mensaje de estado que indica el estado del modelo, por ejemplo si el entrenamiento se ha completado o ha fallado.



Si utiliza el AWS SDK, puede comprobar si se ha producido un error en el archivo de manifiesto de terminal o un error en el contenido del manifiesto de terminal comprobando la respuesta de. DescribeProjectVersions En este caso, el valor de `Status` es `TRAINING_FAILED` y el campo `StatusMessage` contiene el error.

Errores de servicio

Los errores de servicio terminales se producen cuando en Amazon Rekognition se produce un problema con el servicio y no puede continuar con el entrenamiento. Por ejemplo, si falla otro servicio del que dependa Etiquetas personalizadas de Amazon Rekognition. Etiquetas personalizadas de Amazon Rekognition notifica los errores de servicio en la consola cuando se produce un problema de servicio en Amazon Rekognition. Si utilizas el AWS SDK, los errores de servicio que se producen durante el entrenamiento se presentan como `InternalServerError` excepción mediante CreateProjectVersiony DescribeProjectVersions.

Si se produce un error de servicio, vuelva a intentar entrenar el modelo. Si el entrenamiento sigue fallando, póngase en contacto con AWS Support e incluya cualquier información sobre el error que acompañe al error de servicio.

Lista de errores terminales de archivos de manifiesto

Los errores de archivos de manifiesto son terminales, en los conjuntos de datos de entrenamiento y de prueba, que se producen en el propio archivo o en varios archivos. Los errores de archivos de manifiesto se detectan antes de validar el contenido de los conjuntos de datos de entrenamiento y de prueba. Los errores de archivos de manifiesto impiden que se notifiquen errores de validación no terminales. Por ejemplo, un archivo de manifiesto de entrenamiento vacío genera el error The manifest file is empty. Como el archivo está vacío, no se puede informar de ningún error de validación de la línea JSON que no sea terminal. Tampoco se crea el resumen del manifiesto.

Debe corregir los errores del archivo de manifiesto antes de poder entrenar el modelo.

A continuación se indican los errores de los archivos de manifiesto.

- The manifest file extension or contents are invalid.
- El archivo de manifiesto está vacío.
- The manifest file size exceeds the maximum supported size.
- No se puede escribir en el bucket de S3 final.
- Los permisos del bucket de S3 no son los correctos.

Lista de errores terminales de contenido del manifiesto

Los errores de contenido del manifiesto son errores terminales que tienen que ver con el contenido de un manifiesto. Por ejemplo, si aparece el error The manifest file contains insufficient labeled images per label to perform auto-split, el entrenamiento no podrá completarse porque no hay suficientes imágenes etiquetadas en el conjunto de datos de entrenamiento para crear un conjunto de datos de prueba.

Además de aparecer en la consola y en la respuesta de `DescribeProjectVersions`, el error figurará en el resumen del manifiesto junto con cualquier otro error terminal de contenido del manifiesto. Para obtener más información, consulte Qué es el resumen del manifiesto.

Los errores no terminales en las líneas JSON también se indican en manifiestos separados en los resultados de las validaciones de entrenamiento y de prueba. Los errores no terminales en las líneas

JSON detectados por Etiquetas personalizadas de Amazon Rekognition no están necesariamente relacionados con los errores de contenido del manifiesto que hacen detener el entrenamiento. Para obtener más información, consulte Qué son los manifiestos de resultados de validación de entrenamiento y de prueba.

Debe corregir los errores de contenido del manifiesto antes de poder entrenar el modelo.

A continuación se muestran los mensajes de error relacionados con los errores de contenido del manifiesto.

- The manifest file contains too many invalid rows.
- The manifest file contains images from multiple S3 buckets.
- Invalid owner id for images S3 bucket.
- The manifest file contains insufficient labeled images per label to perform auto-split.
- The manifest file has too few labels.
- The manifest file has too many labels.
- Less than {}% label overlap between the training and testing manifest files.
- The manifest file has too few usable labels.
- Less than {}% usable label overlap between the training and testing manifest files.
- Failed to copy images from S3 bucket.

Lista de errores no terminales de validación en líneas JSON

Los errores de validación de las líneas JSON son errores no terminales que no requieren que Etiquetas personalizadas de Amazon Rekognition dejen de entrenar un modelo.

Los errores de validación de las líneas JSON no aparecen en la consola.

En los conjuntos de datos de entrenamiento y de prueba, una línea JSON representa la información de entrenamiento o de prueba de una sola imagen. Los errores de validación en una línea JSON, como una imagen no válida, se indican en los manifiestos de validación de entrenamiento y de prueba. Etiquetas personalizadas de Amazon Rekognition completa el entrenamiento con las demás líneas JSON válidas que se encuentren en el manifiesto. Para obtener más información, consulte Qué son los manifiestos de resultados de validación de entrenamiento y de prueba. Para obtener información sobre las reglas de validación, consulte Reglas de validación de archivos de manifiesto.

> **Note**
>
> El entrenamiento no se realiza si hay demasiados errores en las líneas JSON.

Le recomendamos que corrija también los errores no terminales de las líneas JSON, ya que pueden provocar errores en el futuro o afectar al entrenamiento del modelo.

Etiquetas personalizadas de Amazon Rekognition puede generar los siguientes errores no terminales de validación en las líneas JSON.

- The source-ref key is missing.
- The format of the source-ref value is invalid.
- No label attributes found.
- El formato del atributo de etiqueta {} no es válido.
- The format of the label attributemetadata is invalid.
- No valid label attributes found.
- One or more bounding boxes has a missing confidence value.
- One of more class ids is missing from the class map.
- The JSON Line has an invalid format.
- The image is invalid. Check S3 path and/or image properties.
- The bounding box has off frame values.
- The height and width of the bounding box is too small.
- There are more bounding boxes than the allowed maximum.
- No valid annotations found.

Qué es el resumen del manifiesto

El resumen del manifiesto incluye la siguiente información.

- Información del error de Lista de errores terminales de contenido del manifiesto detectado durante la validación.
- Información sobre la ubicación del error de Lista de errores no terminales de validación en líneas JSON en los conjuntos de datos de entrenamiento y de prueba.

- Estadísticas del error, como el número total de líneas JSON no válidas encontradas en los conjuntos de datos de entrenamiento y de prueba.

El resumen del manifiesto se crea durante el entrenamiento si no hay Lista de errores terminales de archivos de manifiesto. Para obtener la ubicación del archivo del resumen del manifiesto (manifest_summary.json), consulte Cómo obtener los resultados de la validación.

> **Note**
>
> Los errores de servicio y los errores del archivo de manifiesto no vienen incluidos en el resumen del manifiesto. Para obtener más información, consulte Errores terminales.

Para obtener información sobre errores específicos en el contenido del manifiesto, consulte Errores terminales de contenido del manifiesto.

Formato de archivo del resumen del manifiesto

Un archivo de manifiesto cuenta con dos secciones, `statistics` y `errors`.

statistics

`statistics`: contiene información sobre los errores en los conjuntos de datos de entrenamiento y de prueba.

- `training`: estadísticas y errores encontrados en el conjunto de datos de entrenamiento.
- `testing`: estadísticas y errores encontrados en el conjunto de datos de prueba.

Los objetos de la matriz `errors` contienen el código y el mensaje de error del contenido del manifiesto.

La matriz `error_line_indices` contiene los números de línea de cada línea JSON del manifiesto de entrenamiento o de prueba que tiene un error. Para obtener más información, consulte Soluciones de errores de entrenamiento.

errores

Errores que engloban tanto el conjunto de datos de entrenamiento como el de prueba. Por ejemplo, el error ERROR_INSUFICIENT_USABLE_LABEL_OVERLAP se produce cuando no hay suficientes etiquetas utilizables que se superpongan a los conjuntos de datos de entrenamiento y de prueba.

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

Ejemplo de resumen del manifiesto

El siguiente ejemplo es un resumen parcial del manifiesto donde aparece un error terminal en el contenido del manifiesto (ERROR_TOO_MANY_INVALID_ROWS_IN_MANIFEST). La matriz `error_json_line_indices` contiene los números de línea de los errores de línea JSON que no son terminales en el manifiesto de validación de entrenamiento o de prueba correspondiente.

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

Qué son los manifiestos de resultados de validación de entrenamiento y de prueba

Durante el entrenamiento, Etiquetas personalizadas de Amazon Rekognition crea manifiestos de resultados de validación que registran los errores no terminales de líneas JSON. Los manifiestos de resultados de validación son copias de los conjuntos de datos de entrenamiento y de prueba a los que se agrega información sobre los errores. Puede acceder a los manifiestos de validación una vez completado el entrenamiento. Para obtener más información, consulte Cómo obtener los resultados de la validación. Etiquetas personalizadas de Amazon Rekognition también crea un resumen del manifiesto que incluye información general sobre los errores en las líneas JSON,

como las ubicaciones de los errores y el número de errores de las líneas JSON. Para obtener más información, consulte Qué es el resumen del manifiesto.

> **Note**
>
> Los resultados de la validación (los manifiestos de resultados de validación y el resumen del manifiesto del entrenamiento y de las pruebas) solo se crean si no hay Lista de errores terminales de archivos de manifiesto.

Un manifiesto contiene las líneas JSON de cada imagen en el conjunto de datos. En los manifiestos de resultados de validación, la información sobre los errores de las líneas JSON se agrega a las líneas JSON en las que se registran errores.

Un error de línea JSON es un error no terminal relacionado con una sola imagen. Un error de validación no terminal puede invalidar toda la línea JSON o solo una parte. Por ejemplo, si la imagen a la que se hace referencia en una línea JSON no está en formato PNG o JPG, se produce el error ERROR_INVALID_IMAGE y toda la línea JSON queda excluida del entrenamiento. El entrenamiento continuará con las otras líneas JSON válidas.

Dentro de una línea JSON, un error podría significar que la línea JSON todavía se puede usar para el entrenamiento. Por ejemplo, si el valor de la izquierda de uno de los cuatro cuadros delimitadores asociados a una etiqueta es negativo, el modelo seguirá entrenándose con los demás cuadros delimitadores válidos. Se devolverá la información del error de la línea JSON con respecto al cuadro delimitador no válido (ERROR_INVALID_BOUNDING_BOX). En este ejemplo, la información del error se agrega al objeto `annotation` en el que se produce el error.

Los errores de advertencia, como WARNING_NO_ANNOTATIONS, no se utilizan para el entrenamiento y se consideran líneas JSON ignoradas (`ignored_json_lines`) en el resumen del manifiesto. Para obtener más información, consulte Qué es el resumen del manifiesto. Además, las líneas JSON ignoradas no se tienen en cuenta para el umbral de error del 20 % en el entrenamiento y las pruebas.

Para obtener información sobre errores específicos de validación de datos no terminales, consulteErrores no terminales de validación en líneas JSON.

> **Note**
>
> Si hay demasiados errores de validación de datos, se detendrá el entrenamiento y se registrará el error terminal ERROR_TOO_MANY_INVALID_ROWS_IN_MANIFEST en el resumen del manifiesto.

Para obtener información sobre cómo corregir los errores en las líneas JSON, consulte Soluciones de errores de entrenamiento.

Formato de error de líneas JSON

Etiquetas personalizadas de Amazon Rekognition agrega información sobre los errores no terminales de validación relacionados con las imágenes y el formato de la localización de objetos. Para obtener más información, consulte the section called “Creación de un archivo de manifiesto”.

Errores relacionados con imágenes

En el siguiente ejemplo se muestran las matrices `Error` en una línea JSON de imagen. Hay dos tipos de errores. Errores relacionados con los metadatos de los atributos de etiqueta (en este ejemplo, sport-metadata) y errores relacionados con la imagen. El error incluye un código de error (code) y un mensaje de error (message). Para obtener más información, consulte Importación de etiquetas de imagen en los archivos de manifiesto.

```
{
    "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
        }
    ]
}
```

Errores de localización de objetos

En el siguiente ejemplo se muestran las matrices de errores en una línea JSON de localización de objetos. La línea JSON contiene una matriz `Errors` de información sobre los campos de las siguientes secciones de la línea JSON. Cada objeto `Error` incluye el código de error y el mensaje de error.

- label attribute: errores en los campos de atributos de etiqueta. Consulte `bounding-box` en el ejemplo.
- annotations: los errores de anotaciones (cuadros delimitadores) se almacenan en la matriz `annotations` dentro del atributo de etiqueta.
- label attribute-metadata: errores en los metadatos de atributos de etiqueta. Consulte `bounding-box-metadata` en el ejemplo.
- image: errores no relacionados con los campos de atributos de etiqueta, anotaciones ni metadatos de atributos de etiqueta.

Para obtener más información, consulte Localización de objetos en archivos de manifiesto.

```
{
    "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
        }
    ]
}
```

Ejemplo de error en línea JSON

En la siguiente línea JSON de localización de objetos (cuyo formato se ha modificado para facilitar su lectura) aparece el error ERROR_BOUNDING_BOX_TOO_SMALL. En este ejemplo, las dimensiones del cuadro delimitador (alto y ancho) no son superiores a 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"
    }
}
```

Cómo obtener los resultados de la validación

Los resultados de la validación contienen la información de error de Lista de errores terminales de contenido del manifiesto yLista de errores no terminales de validación en líneas JSON. Hay tres archivos de resultados de validación.

- training_manifest_with_validation.json: se agrega una copia del archivo de manifiesto del conjunto de datos de entrenamiento con información sobre los errores en las líneas JSON.
- testing_manifest_with_validation.json: se agrega una copia del archivo de manifiesto del conjunto de datos de prueba con información sobre los errores en las líneas JSON.
- manifest_summary.json: resumen de los errores de contenido del manifiesto y los errores en las líneas de JSON encontrados en los conjuntos de datos de entrenamiento y de prueba. Para obtener más información, consulte Qué es el resumen del manifiesto.

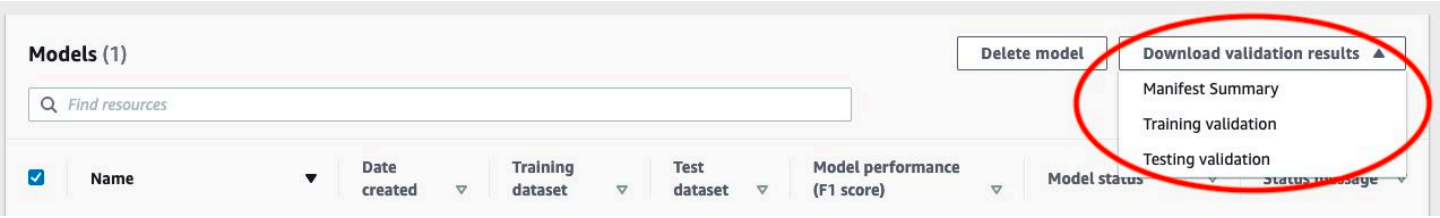
Para obtener información sobre el contenido de los manifiestos de validación del entrenamiento y las pruebas, consulte Depuración de un modelo de entrenamiento con errores.

> **Note**
>
> - Los resultados de la validación se crean solo si no se genera Lista de errores terminales de archivos de manifiesto durante el entrenamiento.
> - Si se produce un error de servicio después de validar el manifiesto de formación y de pruebas, se crean los resultados de la validación, pero la respuesta DescribeProjectVersionsno incluye las ubicaciones de los archivos de resultados de la validación.

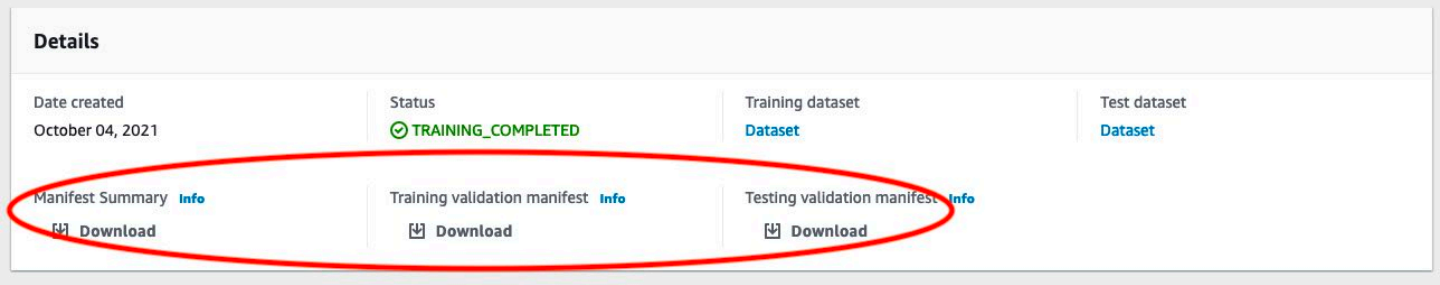
Cuando el entrenamiento finalice o falle, puede descargar los resultados de la validación mediante la consola Amazon Rekognition Custom Labels u obtener la ubicación del bucket de Amazon S3 llamando a la API. DescribeProjectVersions

Cómo obtener los resultados de la validación (consola)

Si utiliza la consola para entrenar el modelo, puede descargar los resultados de la validación en la lista de modelos del proyecto, tal como se muestra en el siguiente diagrama. El panel Modelos muestra el entrenamiento del modelo y los resultados de la validación con la opción de descargarlos.



También puede acceder a la descarga de los resultados de la validación en la página de detalles del modelo. La página de detalles muestra los detalles del conjunto de datos con los conjuntos de datos de estado, entrenamiento y pruebas, y enlaces de descarga para ver el resumen del manifiesto, el manifiesto de validación del entrenamiento y el manifiesto de validación de las pruebas.



Para obtener más información, consulte Entrenamiento de un modelo (consola).

Cómo obtener los resultados de la validación (SDK)

Una vez finalizado el entrenamiento del modelo, Etiquetas personalizadas de Amazon Rekognition almacena los resultados de la validación en el bucket de Amazon S3 especificado durante el entrenamiento. Puede obtener la ubicación del depósito de S3 llamando a la DescribeProjectVersionsAPI, una vez finalizado el entrenamiento. Para entrenar un modelo, consulte Entrenamiento de un modelo (SDK).

Se devuelve un ValidationDataobjeto para el conjunto de datos de entrenamiento (TrainingDataResult) y el conjunto de datos de prueba (TestingDataResult). El resumen del manifiesto se devuelve en `ManifestSummary`.

Una vez que obtenga la ubicación del bucket de Amazon S3, podrá descargar los resultados de la validación. Para obtener más información, consulte ¿Cómo descargo un objeto de un bucket de S3? También puede utilizar la operación GetObject.

Cómo obtener los datos de validación (SDK)

1. Si aún no lo ha hecho, instale y configure el AWS CLI y el AWS SDKs. Para obtener más información, consulte Paso 4: Configure y AWS CLIAWS SDKs.
2. Use el siguiente ejemplo para obtener la ubicación de los resultados de la validación.

   Python

   Sustituya `project_arn` por el nombre de recurso de Amazon (ARN) del proyecto que contiene el modelo. Para obtener más información, consulte Administración de un proyecto de Etiquetas personalizadas de Amazon Rekognition. Sustituya `version_name` por el nombre de la versión del modelo. Para obtener más información, consulte Entrenamiento de un modelo (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. En el resultado final del programa, anote el campo `Validation` dentro de los objetos `TestingDataResult` y `TrainingDataResult`. El resumen del manifiesto está en `ManifestSummary`.

Soluciones de errores de entrenamiento

El resumen del manifiesto sirve para identificar Lista de errores terminales de contenido del manifiesto y Lista de errores no terminales de validación en líneas JSON detectados durante el entrenamiento. Estos errores de contenido del manifiesto se deben corregir. Es recomendable que también resuelva los errores no terminales de las líneas JSON. Para obtener más información sobre errores específicos, consulte Errores no terminales de validación en líneas JSON y Errores terminales de contenido del manifiesto.

Puede hacer correcciones en el conjunto de datos de entrenamiento o de prueba utilizado para el entrenamiento. Asimismo, puede hacer estas correcciones en los archivos de manifiesto de validación del entrenamiento y de las pruebas y utilizarlas para entrenar el modelo.

Después de realizar las correcciones, deberá importar los manifiestos modificados y volver a entrenar el modelo. Para obtener más información, consulte Creación de un archivo de manifiesto.

En el siguiente procedimiento se explica cómo utilizar el resumen del manifiesto para corregir los errores terminales de contenido del manifiesto. El procedimiento también le servirá para detectar y corregir los errores en las líneas JSON de los manifiestos de validación de entrenamiento y de prueba.

Cómo corregir errores de entrenamiento de Etiquetas personalizadas de Amazon Rekognition

1. Descargue los archivos de resultados de la validación. Los nombres de los archivos son training_manifest_with_validation.json, testing_manifest_with_validation.json y manifest_summary.json. Para obtener más información, consulte Cómo obtener los resultados de la validación.
2. Abra el archivo de resumen del manifiesto (manifest_summary.json).
3. Corrija cualquier error en el resumen del manifiesto. Para obtener más información, consulte Qué es el resumen del manifiesto.
4. En el resumen del manifiesto, repita la matriz `error_line_indices` en `training` y subsane los errores en `training_manifest_with_validation.json` en los números de línea JSON correspondientes. Para obtener más información, consulte the section called “Qué son los manifiestos de resultados de validación de entrenamiento y de prueba”.
5. Repita la matriz `error_line_indices` en `testing` y subsane los errores en `testing_manifest_with_validation.json` en los números de línea JSON correspondientes.
6. Vuelva a entrenar el modelo utilizando los archivos de manifiesto de validación como conjuntos de datos de entrenamiento y de prueba. Para obtener más información, consulte the section called “Entrenamiento de un modelo”.

Si utiliza el AWS SDK y decide corregir los errores en los archivos de manifiesto de datos de validación de entrenamiento o de prueba, utilice la ubicación de los archivos de manifiesto de datos de validación en el archivo TrainingDatae TestingDataintroduzca los parámetros para CreateProjectVersion. Para obtener más información, consulte Entrenamiento de un modelo (SDK).

Prioridad de errores en líneas JSON

Primero se detectan los siguientes errores en las líneas JSON. Si se produce alguno de estos errores, se detendrá la validación de los errores de las líneas JSON. Deberá corregir estos errores antes de poder resolver cualquier otro error en las líneas JSON.

- 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

Errores terminales de archivos de manifiesto

En este tema se describe el Lista de errores terminales de archivos de manifiesto. Los errores del archivo de manifiesto no tienen un código de error asociado. Los manifiestos de los resultados de la validación no se crean cuando se produce un error terminal en el archivo de manifiesto. Para obtener más información, consulte Qué es el resumen del manifiesto. Los errores terminales en el manifiesto impiden que se notifiquen Errores no terminales de validación en líneas JSON.

The manifest file extension or contents are invalid.

El archivo de manifiesto de entrenamiento o de prueba no tiene una extensión de archivo o su contenido no es válido.

Cómo corregir el error The manifest file extension or contents are invalid.

- Compruebe las siguientes causas posibles en los archivos de manifiesto de entrenamiento y de prueba.
  - El archivo de manifiesto no tiene la extensión de archivo. Por convención, la extensión del archivo es `.manifest`.
  - No se ha encontrado el bucket de Amazon S3 o la clave del archivo de manifiesto.

El archivo de manifiesto está vacío.

El archivo de manifiesto de entrenamiento o de prueba utilizado para el entrenamiento existe, pero está vacío. El archivo de manifiesto necesita una línea JSON por cada imagen que se utilice para el entrenamiento y las pruebas.

Cómo corregir el error The manifest file is empty.

1. Compruebe cuáles de los manifiestos de entrenamiento o de prueba están vacíos.
2. Añada líneas JSON al archivo de manifiesto vacío. Para obtener más información, consulte Creación de un archivo de manifiesto. También puede crear un conjunto de datos nuevo con la consola. Para obtener más información, consulte the section called “Crear conjuntos de datos con imágenes”.

The manifest file size exceeds the maximum supported size.

El tamaño del archivo de manifiesto de entrenamiento o de prueba (en bytes) es demasiado grande. Para obtener más información, consulte Directrices y cuotas en Etiquetas personalizadas de Amazon Rekognition. Un archivo de manifiesto puede tener menos de la cantidad máxima de líneas JSON y aun así superar el tamaño máximo de archivo.

No puede usar la consola de Etiquetas personalizadas de Amazon Rekognition para corregir el error The manifest file size exceeds the maximum supported size.

Cómo corregir el error The manifest file size exceeds the maximum supported size.

1. Compruebe cuáles de los manifiestos de entrenamiento y de prueba superan el tamaño máximo de archivo.
2. Reduzca el número de líneas JSON demasiado grandes en los archivos de manifiesto. Para obtener más información, consulte Creación de un archivo de manifiesto.

Los permisos del bucket de S3 no son los correctos.

Etiquetas personalizadas de Amazon Rekognition no tiene permisos para uno o varios de los buckets que contienen los archivos de manifiesto de entrenamiento y de prueba.

No es posible usar la consola de Etiquetas personalizadas de Amazon Rekognition para subsanar este error.

Cómo corregir el error The S3 bucket permissions are incorrect.

- Compruebe los permisos de los buckets que contengan los manifiestos de entrenamiento y de prueba. Para obtener más información, consulte Paso 2: Configurar los permisos de la consola de Etiquetas personalizadas de Amazon Rekognition.

No se puede escribir en el bucket de S3 final.

El servicio no puede generar los archivos finales del entrenamiento.

Cómo corregir el error Unable to write to output S3 bucket.

- Compruebe que la información del bucket de Amazon S3 en el parámetro OutputConfigde entrada to CreateProjectVersionsea correcta.

No es posible usar la consola de Etiquetas personalizadas de Amazon Rekognition para subsanar este error.

Errores terminales de contenido del manifiesto

En este tema se describe el error Lista de errores terminales de contenido del manifiesto que se registra en el resumen del manifiesto. El resumen del manifiesto incluye un código de error y un mensaje por cada error detectado. Para obtener más información, consulte Qué es el resumen del manifiesto. Los errores terminales en el contenido del manifiesto no impiden que se notifiquen Lista de errores no terminales de validación en líneas JSON.

ERROR_TOO_MANY_INVALID_ROWS_IN_MANIFEST

Mensaje de error

The manifest file contains too many invalid rows.

Más información

El error `ERROR_TOO_MANY_INVALID_ROWS_IN_MANIFEST` se produce si hay demasiadas líneas JSON que incluyen contenido no válido.

No puede usar la consola de Etiquetas personalizadas de Amazon Rekognition para subsanar el error `ERROR_TOO_MANY_INVALID_ROWS_IN_MANIFEST`.

Cómo corregir ERROR_TOO_MANY_INVALID_ROWS_IN_MANIFEST

1. Compruebe si hay errores de líneas JSON en el manifiesto. Para obtener más información, consulte Qué son los manifiestos de resultados de validación de entrenamiento y de prueba.
2. Cómo corregir las líneas JSON que tienen errores. Para obtener más información, consulte Errores no terminales de validación en líneas JSON.

ERROR_IMAGES_IN_MULTIPLE_S3_BUCKETS

Mensaje de error

The manifest file contains images from multiple S3 buckets.

Más información

Un manifiesto solo puede hacer referencia a las imágenes almacenadas en un único bucket. Cada línea JSON almacena la ubicación en Amazon S3 de una ubicación de imagen con el valor de`source-ref`. En el siguiente ejemplo, el nombre del bucket es my-bucket.

```
"source-ref": "s3://my-bucket/images/sunrise.png"
```

No es posible usar la consola de Etiquetas personalizadas de Amazon Rekognition para subsanar este error.

Cómo corregir **`ERROR_IMAGES_IN_MULTIPLE_S3_BUCKETS`**

- Asegúrese de que todas las imágenes estén en el mismo bucket de Amazon S3 y de que el valor de `source-ref` en cada línea JSON haga referencia al bucket en el que se almacenan las imágenes. Si lo prefiere, elija un bucket de Amazon S3 que prefiera y elimine las líneas JSON en las que `source-ref` no haga referencia a su bucket preferido.

ERROR_INVALID_PERMISSIONS_IMAGES_S3_BUCKET

Mensaje de error

The permissions for the images S3 bucket are invalid.

Más información

Los permisos del bucket de Amazon S3 que contiene las imágenes no son correctos.

No puede usar la consola de Etiquetas personalizadas de Amazon Rekognition para subsanar este error.

Cómo corregir **`ERROR_INVALID_PERMISSIONS_IMAGES_S3_BUCKET`**

- Compruebe los permisos del bucket que contiene las imágenes. El valor de `source-ref` de una imagen incluye la ubicación del bucket.

ERROR_INVALID_IMAGES_S3_BUCKET_OWNER

Mensaje de error

Invalid owner id for images S3 bucket.

Más información

El propietario del bucket que contiene las imágenes de entrenamiento o de prueba es diferente del propietario del bucket que contiene el manifiesto de entrenamiento o de prueba. Puede utilizar el siguiente comando para buscar el propietario del bucket.

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

El `OWNER ID` debe coincidir con los buckets que almacenan las imágenes y los archivos de manifiesto.

Cómo corregir ERROR_INVALID_IMAGES_S3_BUCKET_OWNER

1. Elija al propietario deseado de los buckets de entrenamiento, de prueba, de resultados y de imágenes. El propietario debe tener los permisos para usar Etiquetas personalizadas de Amazon Rekognition.
2. Por cada bucket que actualmente no pertenezca al propietario deseado, cree un nuevo bucket de Amazon S3 que pertenezca a su propietario preferido.
3. Copie el contenido del bucket anterior en el bucket nuevo. Para obtener más información, consulte ¿Cómo puedo copiar objetos entre buckets de Amazon S3?

No puede usar la consola de Etiquetas personalizadas de Amazon Rekognition para subsanar este error.

ERROR_INSUFFICIENT_IMAGES_PER_LABEL_FOR_AUTOSPLIT

Mensaje de error

The manifest file contains insufficient labeled images per label to perform auto-split.

Más información

Durante el entrenamiento del modelo, puede crear un conjunto de datos de prueba utilizando el 20 % de las imágenes del conjunto de datos de entrenamiento. El error ERROR_INSUFFICIENT_IMAGES_PER_LABEL_FOR_AUTOSPLIT se produce cuando no hay suficientes imágenes para crear un conjunto de datos de prueba aceptable.

No puede usar la consola de Etiquetas personalizadas de Amazon Rekognition para subsanar este error.

Cómo corregir ERROR_INSUFFICIENT_IMAGES_PER_LABEL_FOR_AUTOSPLIT

- Añada más imágenes etiquetadas al conjunto de datos de entrenamiento. Puede agregar imágenes en la consola de Etiquetas personalizadas de Amazon Rekognition si agrega imágenes al conjunto de datos de entrenamiento o líneas JSON al manifiesto de entrenamiento. Para obtener más información, consulte Administración de conjuntos de datos.

ERROR_MANIFEST_TOO_FEW_LABELS

Mensaje de error

The manifest file has too few labels.

Más información

Los conjuntos de datos de entrenamiento y de prueba tienen un número mínimo de etiquetas obligatorio. El mínimo depende de si el conjunto de datos entrena o prueba un modelo para detectar etiquetas de imagen (clasificación) o si el modelo detecta la ubicación de objetos. Si el conjunto de datos de entrenamiento se divide para crear un conjunto de datos de prueba, el número de

etiquetas en el conjunto de datos se determinará después de que se divida este. Para obtener más información, consulte Directrices y cuotas en Etiquetas personalizadas de Amazon Rekognition.

Cómo corregir ERROR_MANIFEST_TOO_FEW_LABELS (consola)

1. Añada más etiquetas nuevas al conjunto de datos. Para obtener más información, consulte Administración de etiquetas.
2. Añada las nuevas etiquetas a las imágenes del conjunto de datos. Si el modelo detecta etiquetas de imagen, consulte Asignación de etiquetas de imagen a una imagen. Si el modelo detecta la ubicación de objetos, consulte the section called “Etiquetado de objetos con cuadros delimitadores”.

Cómo corregir ERROR_MANIFEST_TOO_FEW_LABELS (línea JSON)

- Añada líneas JSON de las imágenes nuevas que tengan etiquetas nuevas. Para obtener más información, consulte Creación de un archivo de manifiesto. Si el modelo detecta etiquetas de imagen, agregue nuevos nombres de etiquetas al campo `class-name`. Por ejemplo, la etiqueta de la siguiente imagen es Amanecer.

```
{
    "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"
    }
}
```

  Si el modelo detecta ubicaciones de objetos, agregue etiquetas nuevas al `class-map`, tal como se muestra en el siguiente ejemplo.

```
{
 "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"
  }
}
```

Debe asignar la tabla del mapa de clases a las anotaciones de los cuadros delimitadores. Para obtener más información, consulte Localización de objetos en archivos de manifiesto.

ERROR_MANIFEST_TOO_MANY_LABELS

Mensaje de error

The manifest file has too many labels.

Más información

El número de etiquetas únicas en el manifiesto (conjunto de datos) supera el límite permitido. Si el conjunto de datos de entrenamiento se divide para crear un conjunto de datos de prueba, el número de etiquetas se determinará después de la división.

Cómo corregir ERROR_MANIFEST_TOO_MANY_LABELS (consola)

- Elimine las etiquetas del conjunto de datos. Para obtener más información, consulte Administración de etiquetas. Las etiquetas se eliminan automáticamente de las imágenes y los cuadros delimitadores en el conjunto de datos.

Cómo corregir ERROR_MANIFEST_TOO_MANY_LABELS (línea JSON)

- Manifiestos con líneas JSON de imagen: si la imagen tiene una sola etiqueta, elimine las líneas JSON de las imágenes que utilicen la etiqueta deseada. Si la línea JSON contiene varias etiquetas, elimine solo el objeto JSON de la etiqueta deseada. Para obtener más información, consulte Cómo agregar varias etiquetas de imagen a una imagen.

  Manifiestos con líneas JSON con la ubicación del objeto: elimine el cuadro delimitador y la información de etiqueta asociada a la etiqueta que desee eliminar. Haga esto en cada línea JSON que contenga la etiqueta deseada. Es necesario eliminar la etiqueta de la matriz `class-map` y los objetos correspondientes de la matriz `objects` y `annotations`. Para obtener más información, consulte Localización de objetos en archivos de manifiesto.

ERROR_INSUFICIENT_LABEL_OVERLAP

Mensaje de error

Less than {}% label overlap between the training and testing manifest files.

Más información

Hay menos del 50 % de superposición entre los nombres de las etiquetas del conjunto de datos de prueba y los nombres de las etiquetas del conjunto de datos de entrenamiento.

Cómo corregir ERROR_INSUFFICIENT_LABEL_OVERLAP (consola)

- Elimine las etiquetas del conjunto de datos de entrenamiento. También puede agregar etiquetas más comunes al conjunto de datos de prueba. Para obtener más información, consulte Administración de etiquetas. Las etiquetas se eliminan automáticamente de las imágenes y los cuadros delimitadores en el conjunto de datos.

Cómo corregir ERROR_INSUFFICIENT_LABEL_OVERLAP eliminando las etiquetas del conjunto de datos de entrenamiento (línea JSON)

- Manifiestos con líneas JSON de imagen: si la imagen tiene una sola etiqueta, elimine la línea JSON de la imagen que utilice la etiqueta deseada. Si la línea JSON contiene varias etiquetas, elimine solo el objeto JSON de la etiqueta deseada. Para obtener más información, consulte Cómo agregar varias etiquetas de imagen a una imagen. Haga esto en cada línea JSON del manifiesto que contenga la etiqueta que desee eliminar.

  Manifiestos con líneas JSON con la ubicación del objeto: elimine el cuadro delimitador y la información de etiqueta asociada a la etiqueta que desee eliminar. Haga esto en cada línea JSON que contenga la etiqueta deseada. Es necesario eliminar la etiqueta de la matriz `class-map` y los objetos correspondientes de la matriz `objects` y `annotations`. Para obtener más información, consulte Localización de objetos en archivos de manifiesto.

Cómo corregir ERROR_INSUFFICIENT_LABEL_OVERLAP añadiendo etiquetas comunes al conjunto de datos de prueba (línea JSON)

- Añada líneas JSON al conjunto de datos de prueba que incluyan imágenes etiquetadas con etiquetas que ya estén en el conjunto de datos de entrenamiento. Para obtener más información, consulte Creación de un archivo de manifiesto.

ERROR_MANIFEST_TOO_FEW_USABLE_LABELS

Mensaje de error

The manifest file has too few usable labels.

Más información

Un manifiesto de entrenamiento puede contener líneas JSON en formato de etiqueta de imagen y en formato de ubicación de objetos. Según el tipo de líneas JSON que se encuentren en el manifiesto de entrenamiento, Etiquetas personalizadas de Amazon Rekognition creará un modelo que detecta las etiquetas de imagen o un modelo que detecte la ubicación de los objetos. Etiquetas personalizadas de Amazon Rekognition filtra los registros de JSON válidos de las líneas JSON que no están en el formato elegido. ERROR_MANIFEST_TOO_FEW_USABLE_LABELS se produce cuando el número de etiquetas del manifiesto en el tipo de modelo elegido es insuficiente para entrenar el modelo.

Se necesita un mínimo de 1 etiqueta para entrenar un modelo que detecte etiquetas de imagen. Se necesita un mínimo de 2 etiquetas para entrenar un modelo que ubique los objetos.

Cómo corregir ERROR_MANIFEST_TOO_FEW_USABLE_LABELS (consola)

1. Marque el campo `use_case` en el resumen del manifiesto.
2. Añada más etiquetas al conjunto de datos de entrenamiento en este caso (de imagen o localización de objetos) que coincidan con el valor de `use_case`. Para obtener más información, consulte Administración de etiquetas. Las etiquetas se eliminan automáticamente de las imágenes y los cuadros delimitadores en el conjunto de datos.

Cómo corregir ERROR_MANIFEST_TOO_FEW_USABLE_LABELS (línea JSON)

1. Marque el campo `use_case` en el resumen del manifiesto.
2. Añada más etiquetas al conjunto de datos de entrenamiento en este caso (de imagen o localización de objetos) que coincidan con el valor de `use_case`. Para obtener más información, consulte Creación de un archivo de manifiesto.

ERROR_INSUFICIENT_USABLE_LABEL_OVERLAP

Mensaje de error

Less than {}% usable label overlap between the training and testing manifest files.

Más información

Un manifiesto de entrenamiento puede contener líneas JSON en formato de etiqueta de imagen y en formato de ubicación de objetos. Según los formatos que haya en el manifiesto de entrenamiento, Etiquetas personalizadas de Amazon Rekognition creará un modelo que detecta las etiquetas de imagen o un modelo que detecte la ubicación de los objetos. Etiquetas personalizadas de Amazon Rekognition no usará registros de JSON válidos de las líneas JSON que no están en el formato elegido. ERROR_INSUFFICIENT_USABLE_LABEL_OVERLAP se produce cuando hay menos del 50 % de superposición entre las etiquetas de prueba y de entrenamiento que se utilizan.

Cómo corregir ERROR_INSUFFICIENT_USABLE_LABEL_OVERLAP (consola)

- Elimine las etiquetas del conjunto de datos de entrenamiento. También puede agregar etiquetas más comunes al conjunto de datos de prueba. Para obtener más información, consulte Administración de etiquetas. Las etiquetas se eliminan automáticamente de las imágenes y los cuadros delimitadores en el conjunto de datos.

Cómo corregir ERROR_INSUFFICIENT_USABLE_LABEL_OVERLAP eliminando las etiquetas del conjunto de datos de entrenamiento (línea JSON)

- Conjuntos de datos utilizados para detectar etiquetas de imagen: si la imagen tiene una sola etiqueta, elimine la línea JSON de la imagen que use la etiqueta deseada. Si la línea JSON contiene varias etiquetas, elimine solo el objeto JSON de la etiqueta deseada. Para obtener más información, consulte Cómo agregar varias etiquetas de imagen a una imagen. Haga esto en cada línea JSON del manifiesto que contenga la etiqueta que desee eliminar.

  Conjuntos de datos usados para detectar ubicaciones de objetos: elimine el cuadro delimitador y la información de etiqueta asociada a la etiqueta que desee eliminar. Haga esto en cada línea JSON que contenga la etiqueta deseada. Es necesario eliminar la etiqueta de la matriz `class-map` y los objetos correspondientes de la matriz `objects` y `annotations`. Para obtener más información, consulte Localización de objetos en archivos de manifiesto.

Cómo corregir ERROR_INSUFFICIENT_USABLE_LABEL_OVERLAP añadiendo etiquetas comunes al conjunto de datos de prueba (línea JSON)

- Añada líneas JSON al conjunto de datos de prueba que incluyan imágenes etiquetadas con etiquetas que ya estén en el conjunto de datos de entrenamiento. Para obtener más información, consulte Creación de un archivo de manifiesto.

ERROR_FAILED_IMAGES_S3_COPY

Mensaje de error

Failed to copy images from S3 bucket.

Más información

El servicio no ha podido copiar ninguna de las imágenes en el conjunto de datos.

No puede usar la consola de Etiquetas personalizadas de Amazon Rekognition para subsanar este error.

Cómo corregir ERROR_FAILED_IMAGES_S3_COPY

1. Compruebe los permisos de las imágenes.
2. Si lo está utilizando AWS KMS, consulte la política de buckets. Para obtener más información, consulte Descifrar archivos cifrados con AWS Key Management Service.

El archivo de manifiesto tiene demasiados errores terminales.

Hay demasiadas líneas JSON con errores terminales en el contenido.

Cómo corregir **`ERROR_TOO_MANY_RECORDS_IN_ERROR`**

- Reduzca el número de líneas JSON (imágenes) con errores terminales en el contenido. Para obtener más información, consulte Errores terminales de contenido del manifiesto.

No puede usar la consola de Etiquetas personalizadas de Amazon Rekognition para subsanar este error.

Errores no terminales de validación en líneas JSON

En este tema se indican los errores no terminales de validación de líneas JSON notificados por Etiquetas personalizadas de Amazon Rekognition durante el entrenamiento. Los errores vienen recogidos en el manifiesto de validación de entrenamiento y de prueba. Para obtener más información, consulte Qué son los manifiestos de resultados de validación de entrenamiento y de prueba. Puede corregir un error no terminal de línea JSON modificando la línea JSON en el archivo de manifiesto de entrenamiento o de prueba. También puede eliminar la línea JSON del manifiesto, pero hacerlo podría reducir la calidad del modelo. Si hay muchos errores de validación no terminales, puede que le resulte más fácil volver a crear el archivo de manifiesto. Los errores de validación suelen producirse en los archivos de manifiesto creados manualmente. Para obtener más información, consulte Creación de un archivo de manifiesto. Para obtener información sobre cómo resolver errores de validación, consulte Soluciones de errores de entrenamiento. Algunos errores se pueden subsanar en la consola de Etiquetas personalizadas de Amazon Rekognition.

ERROR_MISSING_SOURCE_REF

Mensaje de error

The source-ref key is missing.

Más información

El campo `source-ref` de la línea JSON da la ubicación de Amazon S3 de una imagen. Este error se produce cuando falta la clave de `source-ref` o está mal escrita. Este error suele producirse en los archivos de manifiesto creados manualmente. Para obtener más información, consulte Creación de un archivo de manifiesto.

Cómo corregir **`ERROR_MISSING_SOURCE_REF`**

1. Compruebe que la clave de `source-ref` esté presente y que esté escrita correctamente. La clave y el valor de `source-ref` completos son similares a lo siguiente: `"source-ref": "s3://bucket/path/image"`.
2. Modifique la clave de `source-ref` en la línea JSON. También puede eliminar la línea JSON del archivo de manifiesto.

No puede usar la consola de Etiquetas personalizadas de Amazon Rekognition para subsanar este error.

ERROR_INVALID_SOURCE_REF_FORMAT

Mensaje de error

The format of the source-ref value is invalid.

Más información

La clave de `source-ref` está presente en la línea JSON, pero el esquema de la ruta de Amazon S3 es incorrecto. Por ejemplo, la ruta es `https://....` en lugar de `S3://....`. Por lo general, se produce el error ERROR_INVALID_SOURCE_REF_FORMAT en los archivos de manifiesto creados manualmente. Para obtener más información, consulte Creación de un archivo de manifiesto.

Cómo corregir **`ERROR_INVALID_SOURCE_REF_FORMAT`**

1. Compruebe que el esquema sea `"source-ref": "s3://bucket/path/image"`. Por ejemplo, `"source-ref": "s3://custom-labels-console-us-east-1-1111111111/images/000000242287.jpg"`.
2. Modifique o elimine la línea JSON en el archivo de manifiesto.

No puede usar la consola de Etiquetas personalizadas de Amazon Rekognition para subsanar el error `ERROR_INVALID_SOURCE_REF_FORMAT`.

ERROR_NO_LABEL_ATTRIBUTES

Mensaje de error

No label attributes found.

Más información

El atributo de etiqueta o el nombre de clave `-metadata` del atributo de etiqueta (o ambos) no son válidos o no están disponibles. En el siguiente ejemplo, `ERROR_NO_LABEL_ATTRIBUTES` aparece siempre que falta la clave de `bounding-box` o `bounding-box-metadata` (o ambas). Para obtener más información, consulte Creación de un archivo de manifiesto.

```
{
 "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"
  }
}
```

El error `ERROR_NO_LABEL_ATTRIBUTES` suele aparecer en un archivo de manifiesto creado manualmente. Para obtener más información, consulte Creación de un archivo de manifiesto.

Cómo corregir **`ERROR_NO_LABEL_ATTRIBUTES`**

1. Compruebe que el identificador del atributo de etiqueta y las claves de `-metadata` de los identificadores del atributo de etiqueta estén presentes y que los nombres de las claves estén escritos correctamente.

2. Modifique o elimine la línea JSON en el archivo de manifiesto.

No puede usar la consola de Etiquetas personalizadas de Amazon Rekognition para subsanar este error.

ERROR_INVALID_LABEL_ATTRIBUTE_FORMAT

Mensaje de error

The format of the label attribute {} is invalid.

Más información

Falta el esquema de la clave del atributo de etiqueta o no es válido. Por lo general, se produce el error ERROR_INVALID_LABEL_ATTRIBUTE_FORMAT en los archivos de manifiesto creados manualmente. Para obtener más información, consulte Creación de un archivo de manifiesto.

Cómo corregir **`ERROR_INVALID_LABEL_ATTRIBUTE_FORMAT`**

1. Compruebe que la sección de la línea JSON de la clave del atributo de etiqueta sea correcta. En el siguiente ejemplo de ubicación de objetos, los objetos `image_size` y `annotations` deben ser correctos. La clave del atributo de etiqueta se llama `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. Modifique o elimine la línea JSON en el archivo de manifiesto.

No puede usar la consola de Etiquetas personalizadas de Amazon Rekognition para subsanar este error.

ERROR_INVALID_LABEL_ATTRIBUTE_METADATA_FORMAT

Mensaje de error

The format of the label attribute metadata is invalid.

Más información

Falta el esquema de la clave de los metadatos del atributo de etiqueta o no son válidos. Por lo general, se produce el error ERROR_INVALID_LABEL_ATTRIBUTE_METADATA_FORMAT en los archivos de manifiesto creados manualmente. Para obtener más información, consulte Creación de un archivo de manifiesto.

Cómo corregir **`ERROR_INVALID_LABEL_ATTRIBUTE_FORMAT`**

1. Compruebe que el esquema de líneas JSON con la clave de metadatos del atributo de etiqueta es similar al del siguiente ejemplo. La clave de los metadatos del atributo de etiqueta se llama `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. Modifique o elimine la línea JSON en el archivo de manifiesto.

No puede usar la consola de Etiquetas personalizadas de Amazon Rekognition para subsanar este error.

ERROR_NO_VALID_LABEL_ATTRIBUTES

Mensaje de error

No valid label attributes found.

Más información

No se han encontrado atributos de etiqueta válidos en la línea JSON. Etiquetas personalizadas de Amazon Rekognition comprueba tanto el atributo de etiqueta como su identificador. Por lo general, se produce el error ERROR_INVALID_LABEL_ATTRIBUTE_FORMAT en los archivos de manifiesto creados manualmente. Para obtener más información, consulte Creación de un archivo de manifiesto.

Si una línea JSON no tiene un formato de manifiesto de SageMaker IA compatible, Amazon Rekognition Custom Labels marca la línea JSON como no válida `ERROR_NO_VALID_LABEL_ATTRIBUTES` y se informa de un error. Actualmente, Etiquetas personalizadas de Amazon Rekognition admite los formatos de tareas de clasificación y cuadros delimitadores. Para obtener más información, consulte Creación de un archivo de manifiesto.

Cómo corregir **`ERROR_NO_VALID_LABEL_ATTRIBUTES`**

1. Compruebe que el JSON de la clave y de los metadatos del atributo de etiqueta sean correctos.
2. Modifique o elimine la línea JSON en el archivo de manifiesto. Para obtener más información, consulte the section called “Creación de un archivo de manifiesto”.

No puede usar la consola de Etiquetas personalizadas de Amazon Rekognition para subsanar este error.

ERROR_MISSING_BOUNDING_BOX_CONFIDENCE

Mensaje de error

One or more bounding boxes has a missing confidence value.

Más información

Falta la clave de confidence en uno o varios cuadros delimitadores de ubicación de objetos. La clave de confidence de un cuadro delimitador se encuentra en los metadatos del atributo de etiqueta, tal como se muestra en el siguiente ejemplo. El error ERROR_MISSING_BOUNDING_BOX_CONFIDENCE suele producirse en los archivos de manifiesto creados manualmente. Para obtener más información, consulte the section called “Localización de objetos en archivos de manifiesto”.

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

Cómo corregir **`ERROR_MISSING_BOUNDING_BOX_CONFIDENCE`**

1. Compruebe que la matriz `objects` del atributo de etiqueta tiene el mismo número de claves de confidence que los objetos de la matriz `annotations` del atributo de etiqueta.
2. Modifique o elimine la línea JSON en el archivo de manifiesto.

No puede usar la consola de Etiquetas personalizadas de Amazon Rekognition para subsanar este error.

ERROR_MISSING_CLASS_MAP_ID

Mensaje de error

One of more class ids is missing from the class map.

Más información

El objeto `class_id` en un objeto de anotación (cuadro delimitador) no tiene ninguna entrada igual en el mapa de clases de metadatos de atributos de etiqueta (`class-map`). Para obtener más información, consulte Localización de objetos en archivos de manifiesto. El error ERROR_MISSING_CLASS_MAP_ID suele producirse en los archivos de manifiesto creados manualmente.

Cómo corregir ERROR_MISSING_CLASS_MAP_ID

1. Compruebe que el valor `class_id` de cada objeto de anotación (cuadro delimitador) tenga un valor correspondiente en la matriz `class-map`, tal como se muestra en el siguiente ejemplo. La matriz `annotations` y la matriz `class_map` deben tener el mismo número de elementos.

```
{
 "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. Modifique o elimine la línea JSON en el archivo de manifiesto.

No puede usar la consola de Etiquetas personalizadas de Amazon Rekognition para subsanar este error.

ERROR_INVALID_JSON_LINE

Mensaje de error

The JSON Line has an invalid format.

Más información

Se ha encontrado un carácter inesperado en la línea JSON. La línea JSON se cambia por una nueva línea JSON que contiene solo la información del error. Por lo general, se produce el error ERROR_INVALID_JSON_LINE en los archivos de manifiesto creados manualmente. Para obtener más información, consulte the section called “Localización de objetos en archivos de manifiesto”.

No puede usar la consola de Etiquetas personalizadas de Amazon Rekognition para subsanar este error.

Cómo corregir **`ERROR_INVALID_JSON_LINE`**

1. Abra el archivo de manifiesto y vaya a la línea JSON en la que está el error ERROR_INVALID_JSON_LINE.
2. Compruebe que la línea JSON no contenga caracteres no válidos y que no falten caracteres `;` o `,` obligatorios.
3. Modifique o elimine la línea JSON en el archivo de manifiesto.

ERROR_INVALID_IMAGE

Mensaje de error

The image is invalid. Revise la ruta de S3 o las propiedades de la imagen.

Más información

El archivo al que hace referencia `source-ref` no es una imagen válida. Entre las posibles causas, está la relación de aspecto de la imagen, el tamaño de la imagen y el formato de la imagen.

Para obtener más información, consulte Directrices y cuotas.

Cómo corregir **`ERROR_INVALID_IMAGE`**

1. Compruebe lo siguiente.
   - La relación de aspecto de la imagen es inferior a 20:1.
   - El tamaño de la imagen es superior a 15 MB.
   - La imagen está en formato PNG o JPEG.
   - La ruta de la imagen en `source-ref` es correcta.
   - La dimensión mínima de la imagen es superior a 64 píxeles x 64 píxeles.
   - La dimensión máxima de la imagen es superior a 4096 píxeles x 4096 píxeles.
2. Modifique o elimine la línea JSON en el archivo de manifiesto.

No puede usar la consola de Etiquetas personalizadas de Amazon Rekognition para subsanar este error.

ERROR_INVALID_IMAGE_DIMENSION

Mensaje de error

The image dimension(s) do not conform to allowed dimensions.

Más información

La imagen a la que hace referencia `source-ref` no se ajusta a las dimensiones de imagen permitidas. La dimensión mínima es de 64 píxeles. La dimensión máxima es de 4096 píxeles. Se generará el error `ERROR_INVALID_IMAGE_DIMENSION` en el caso de las imágenes con cuadros delimitadores.

Para obtener más información, consulte Directrices y cuotas.

Cómo corregir **`ERROR_INVALID_IMAGE_DIMENSION`** (consola)

1. Modifique la imagen del bucket de Amazon S3 con las dimensiones que Etiquetas personalizadas de Amazon Rekognition pueda procesar.
2. En la consola de Etiquetas personalizadas de Amazon Rekognition, haga lo siguiente:
   a. Elimine los cuadros delimitadores existentes de la imagen.

b. Vuelva a agregar los cuadros delimitadores a la imagen.

c. Guarde los cambios.

Para obtener más información, Etiquetado de objetos con cuadros delimitadores.

Cómo corregir **`ERROR_INVALID_IMAGE_DIMENSION`** (SDK)

1. Modifique la imagen del bucket de Amazon S3 con las dimensiones que Etiquetas personalizadas de Amazon Rekognition pueda procesar.
2. Llame para obtener la línea JSON existente para la imagen. ListDatasetEntries En el parámetro de entrada `SourceRefContains`, indique la ubicación de Amazon S3 y el nombre de archivo de la imagen.
3. Llama UpdateDatasetEntriesy proporciona la línea JSON de la imagen. Asegúrese de que el valor de `source-ref` coincide con la ubicación de la imagen en el bucket de Amazon S3. Actualice las anotaciones del cuadro delimitador para que coincidan con las dimensiones del cuadro delimitador necesarias para la nueva imagen modificada.

```
{
 "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

Mensaje de error

The bounding box has off frame values.

Más información

La información del cuadro delimitador especifica una imagen que está fuera del marco de la imagen o que tiene valores negativos.

Para obtener más información, consulte Directrices y cuotas.

Cómo corregir **`ERROR_INVALID_BOUNDING_BOX`**

1. Compruebe los valores de los cuadros delimitadores de la matriz `annotations`.

```
 "bounding-box": {
  "image_size": [{
   "width": 640,
   "height": 480,
   "depth": 3
  }],
  "annotations": [{
   "class_id": 1,
```

```
   "top": 251,
   "left": 399,
   "width": 155,
   "height": 101
  }]
 },
```

2. Actualice o elimine la línea JSON del archivo de manifiesto.

No puede usar la consola de Etiquetas personalizadas de Amazon Rekognition para subsanar este error.

ERROR_NO_VALID_ANNOTATIONS

Mensaje de error

No valid annotations found.

Más información

Ninguno de los objetos de anotación en la línea JSON contiene información válida sobre los cuadros delimitadores.

Cómo corregir **`ERROR_NO_VALID_ANNOTATIONS`**

1. Actualice la matriz `annotations` para incluir objetos de cuadro delimitador válidos. Compruebe también que la información de los cuadros delimitadores correspondiente (`confidence` y `class_map`) en los metadatos de atributos de etiqueta sea correcta. Para obtener más información, consulte Localización de objetos en archivos de manifiesto.

```
{
 "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. Actualice o elimine la línea JSON del archivo de manifiesto.

No puede usar la consola de Etiquetas personalizadas de Amazon Rekognition para subsanar este error.

ERROR_BOUNDING_BOX_TOO_SMALL

Mensaje de error

The height and width of the bounding box is too small.

Más información

Las dimensiones del cuadro delimitador (altura y anchura) deben ser superiores a 1 x 1 píxel.

Durante el entrenamiento, Etiquetas personalizadas de Amazon Rekognition cambia el tamaño de una imagen si alguna de sus dimensiones supera los 1280 píxeles (las imágenes de origen no se ven afectadas). Los valores finales de altura y anchura del cuadro delimitador deben ser superiores a 1 x 1 píxel. La ubicación de un cuadro delimitador se almacena en la matriz `annotations` de la línea JSON de una ubicación de objeto. Para obtener más información, consulte Localización de objetos en archivos de manifiesto

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

La información del error se agrega al objeto de anotación.

Cómo corregir ERROR_BOUNDING_BOX_TOO_SMALL

- Elija una de las siguientes opciones.
  - Aumente el tamaño de los cuadros delimitadores que sean demasiado pequeños.
  - Elimine los cuadros delimitadores que sean demasiado pequeños. Para obtener información sobre cómo eliminar un cuadro delimitador, consulte ERROR_TOO_MANY_BOUNDING_BOXES.
  - Elimine la imagen (línea JSON) del manifiesto.

ERROR_TOO_MANY_BOUNDING_BOXES

Mensaje de error

There are more bounding boxes than the allowed maximum.

Más información

Hay más cuadros delimitadores que el límite permitido (50). Puede eliminar los cuadros delimitadores en la consola de Etiquetas personalizadas de Amazon Rekognition o puede eliminarlos en la línea JSON.

Cómo corregir **`ERROR_TOO_MANY_BOUNDING_BOXES`** (consola)

1. Decida qué cuadros delimitadores quiere eliminar.
2. Abra la consola Amazon Rekognition en. https://console.aws.amazon.com/rekognition/
3. Elija Usar etiquetas personalizadas.
4. Elija Comenzar.
5. En el panel de navegación de la izquierda, elija el proyecto que contenga el conjunto de datos que desea usar.
6. En la página Conjuntos de datos, elija el conjunto de datos que quiera utilizar.
7. En la página de la galería de conjuntos de datos, seleccione Empezar a etiquetar para activar el modo de etiquetado.
8. Elija la imagen de la que desee eliminar de los cuadros delimitadores.
9. Seleccione Dibujar cuadro delimitador.
10. En la herramienta de dibujo, elija el cuadro delimitador que desee eliminar.
11. Pulse la tecla de borrar del teclado para eliminar el cuadro delimitador.
12. Repita los 2 pasos anteriores hasta que haya eliminado los cuadros delimitadores necesarios.
13. Seleccione Listo.
14. Elija Guardar cambios para guardar los cambios.
15. Seleccione Salir para salir del modo de etiquetado.

Cómo corregir ERROR_TOO_MANY_BOUNDING_BOXES (línea JSON).

1. Abra el archivo de manifiesto y vaya a la línea JSON en la que está el error ERROR_TOO_MANY_BOUNDING_BOXES.
2. Elimine lo siguiente en cada cuadro delimitador que desee eliminar.

   - Elimine el objeto `annotation` correspondiente de la matriz `annotations`.
   - Elimine el objeto `confidence` correspondiente de la matriz `objects` en los metadatos del atributo de etiqueta.
   - Si ya no se utiliza en otros cuadros delimitadores, elimine la etiqueta del `class-map`.

   Consulte el siguiente ejemplo para identificar qué elementos debe eliminar.

```
{
 "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

Mensajes de advertencia

Record is unannotated.

Más información

Una imagen añadida a un conjunto de datos a través de la consola de Etiquetas personalizadas de Amazon Rekognition no estaba etiquetada. La línea JSON de la imagen no se usa para el entrenamiento.

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

Cómo corregir WARNING_UNANNOTATED_RECORD

- Etiquete la imagen en la consola de Etiquetas personalizadas de Amazon Rekognition. Para obtener instrucciones, consulte Asignación de etiquetas de imagen a una imagen.

WARNING_NO_ANNOTATIONS

Mensajes de advertencia

No annotations provided.

Más información

Una línea JSON en formato de localización de objetos no contiene la información de ningún cuadro delimitador, a pesar de que una persona ha incluido las anotaciones (`human-annotated = yes`). La línea JSON es válida, pero no se usa para el entrenamiento. Para obtener más información, consulte Qué son los manifiestos de resultados de validación de entrenamiento y de prueba.

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

Cómo corregir WARNING_NO_ANNOTATIONS

- Elija una de las siguientes opciones.
  - Añada la información del cuadro delimitador (`annotations`) a la línea JSON. Para obtener más información, consulte Localización de objetos en archivos de manifiesto.
  - Elimine la imagen (línea JSON) del manifiesto.

WARNING_NO_ATTRIBUTE_ANNOTATIONS

Mensajes de advertencia

No attribute annotations provided.

Más información

Una línea JSON en formato de localización de objetos no contiene la información de anotaciones de ningún cuadro delimitador, a pesar de que una persona ha incluido las anotaciones (`human-annotated = yes`). La matriz `annotations` no está presente o no se ha rellenado. La línea JSON es válida, pero no se usa para el entrenamiento. Para obtener más información, consulte Qué son los manifiestos de resultados de validación de entrenamiento y de prueba.

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

Cómo corregir WARNING_NO_ATTRIBUTE_ANNOTATIONS

- Elija una de las siguientes opciones.
  - Añada uno o varios objetos `annotation` del cuadro delimitador a la línea JSON. Para obtener más información, consulte Localización de objetos en archivos de manifiesto.
  - Elimine el atributo del cuadro delimitador.

- Elimine la imagen (línea JSON) del manifiesto. Si existen otros atributos del cuadro delimitador válidos en la línea JSON, puede eliminar solo el atributo del cuadro delimitador no válido de la línea JSON.

ERROR_UNSUPPORTED_USE_CASE_TYPE

Mensajes de advertencia

Más información

El valor del campo `type` no es `groundtruth/image-classification` ni `groundtruth/object-detection`. Para obtener más información, consulte Creación de un archivo de manifiesto.

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

Cómo corregir ERROR_UNSUPPORTED_USE_CASE_TYPE

- Seleccione una de las siguientes opciones:
  - Cambie el valor del campo `type` por `groundtruth/image-classification` o `groundtruth/object-detection`, según el tipo de modelo que desee crear. Para obtener más información, consulte Creación de un archivo de manifiesto.
  - Elimine la imagen (línea JSON) del manifiesto.

ERROR_INVALID_LABEL_NAME_LENGTH

Más información

La longitud del nombre de una etiqueta es demasiado larga. La longitud máxima es de 256 caracteres.

Cómo corregir ERROR_INVALID_LABEL_NAME_LENGTH

- Seleccione una de las siguientes opciones:
  - Reduzca la longitud del nombre de la etiqueta a 256 caracteres o menos.
  - Elimine la imagen (línea JSON) del manifiesto.

Mejora de un modelo de Etiquetas personalizadas de Amazon Rekognition

Cuando se complete el entrenamiento, se evalúa el rendimiento del modelo. Para ayudarle, Etiquetas personalizadas de Amazon Rekognition genera unas métricas resumidas y métricas de evaluación para cada etiqueta. Para obtener más información sobre las métricas disponibles, consulte Métricas para evaluar su modelo. Para mejorar el modelo mediante métricas, consulte Mejora de un modelo de Etiquetas personalizadas de Amazon Rekognition.

Si está satisfecho con la precisión del modelo, puede empezar a utilizarlo. Para obtener más información, consulte Ejecución de un modelo de Etiquetas personalizadas de Amazon Rekognition.

Temas

Métricas para evaluar su modelo

Una vez entrenado el modelo, Etiquetas personalizadas de Amazon Rekognition devuelve las métricas de las pruebas del modelo, que puede utilizar para evaluar el rendimiento del modelo. En este tema se describen las métricas que tiene a su disposición y cómo saber si su modelo entrenado está funcionando bien.

La consola de Etiquetas personalizadas de Amazon Rekognition genera las siguientes métricas a modo de resumen de los resultados del entrenamiento y en forma de métricas para cada etiqueta:

- Precisión
- Exhaustividad
- F1

Cada métrica que se genere es una métrica de uso común para evaluar el rendimiento de un modelo de Machine Learning. Etiquetas personalizadas de Amazon Rekognition devuelve las métricas de

los resultados de las pruebas de todo el conjunto de datos de prueba, junto con las métricas de cada etiqueta personalizada. También puede revisar el rendimiento del modelo personalizado entrenado por cada imagen del conjunto de datos de prueba. Para obtener más información, consulte Acceso a las métricas de evaluación (consola).

Evaluación del rendimiento de los modelos

Durante las pruebas, Etiquetas personalizadas de Amazon Rekognition predice si una imagen de prueba contiene una etiqueta personalizada. La puntuación de confianza es un valor que cuantifica la certeza de la predicción del modelo.

Si la puntuación de confianza de una etiqueta personalizada supera el valor del umbral, el resultado del modelo incluirá esta etiqueta. Las predicciones se pueden clasificar de las siguientes maneras:

- Verdadero positivo: el modelo de Etiquetas personalizadas de Amazon Rekognition predice correctamente la presencia de la etiqueta personalizada en la imagen de prueba. Es decir, la etiqueta pronosticada también es una etiqueta con “datos reales” en cuanto a esa imagen. Por ejemplo, Etiquetas personalizadas de Amazon Rekognition devuelve correctamente la etiqueta de un balón de fútbol cuando hay uno en una imagen.
- Falso positivo: el modelo de Etiquetas personalizadas de Amazon Rekognition predice incorrectamente la presencia de la etiqueta personalizada en la imagen de prueba. Es decir, la etiqueta pronosticada también no es una etiqueta con «datos reales» en cuanto a esa imagen. Por ejemplo, Etiquetas personalizadas de Amazon Rekognition devuelve la etiqueta de un balón de fútbol, aunque no haya ninguna etiqueta de balón de fútbol en los datos reales de esa imagen.
- Falso negativo: el modelo de Etiquetas personalizadas de Amazon Rekognition no predice la presencia de una etiqueta personalizada en la imagen, pero los “datos reales” de esa imagen incluyen esta etiqueta. Por ejemplo, Etiquetas personalizadas de Amazon Rekognition no devuelve una etiqueta personalizada de “balón de fútbol” en una imagen que contiene un balón de fútbol.
- Verdadero negativo: el modelo de Etiquetas personalizadas de Amazon Rekognition predice correctamente que una etiqueta personalizada no está presente en la imagen de prueba. Por ejemplo, Etiquetas personalizadas de Amazon Rekognition no devuelve la etiqueta personalizada de un balón de fútbol en una imagen que no contiene un balón de fútbol.

La consola da acceso a los valores verdaderos positivos, falsos positivos y falsos negativos de cada imagen del conjunto de datos de prueba. Para obtener más información, consulte Acceso a las métricas de evaluación (consola).

Estos resultados de predicción se utilizan para calcular las siguientes métricas en cada etiqueta y un valor acumulado de todo el conjunto de pruebas. Las mismas definiciones se aplican a las predicciones realizadas por el modelo en cuanto a los cuadros delimitadores, con la diferencia de que todas las métricas se calculan por cada cuadro delimitador (predicción o dato real) en cada imagen de prueba.

Intersección sobre la unión (IoU) y detección de objetos

La intersección sobre la unión (IoU) calcula el porcentaje de superposición entre dos cuadros delimitadores de objetos en su área combinada. El rango va de 0 (superposición más baja) a 1 (superposición completa). Durante las pruebas, un cuadro delimitador predicho es correcto cuando el IoU del cuadro delimitador con el dato real y el cuadro delimitador previsto es de al menos 0,5.

Umbral supuesto

Las Etiquetas personalizadas de Amazon Rekognition calculan automáticamente un valor de umbral supuesto (0-1) para cada una de sus etiquetas personalizadas. No puede definir el valor de umbral supuesto en una etiqueta personalizada. El umbral supuesto de cada etiqueta es el valor por encima del cual una predicción se cuenta como un verdadero positivo o un falso positivo. Se establece en función del conjunto de datos de prueba. El umbral supuesto se calcula en función de la mejor puntuación de F1 obtenida en el conjunto de datos de prueba durante el entrenamiento del modelo.

Puede obtener el valor del umbral supuesto de una etiqueta a partir de los resultados de entrenamiento del modelo. Para obtener más información, consulte Acceso a las métricas de evaluación (consola).

Los cambios en los valores de umbral supuesto se utilizan normalmente para mejorar la precisión y la capacidad de recuperación de un modelo. Para obtener más información, consulte Mejora de un modelo de Etiquetas personalizadas de Amazon Rekognition. Como no se puede establecer el umbral supuesto de un modelo en una etiqueta, se pueden obtener los mismos resultados analizando una imagen con `DetectCustomLabels` e indicando el parámetro de entrada `MinConfidence`. Para obtener más información, consulte Análisis de una imagen con un modelo entrenado.

Precisión

Las Etiquetas personalizadas de Amazon Rekognition generan proporciona métricas de precisión para cada etiqueta y una métrica de precisión media para todo el conjunto de datos de prueba.

La precisión es la fracción de las predicciones correctas (verdaderos positivos) sobre todas las predicciones del modelo (verdaderos positivos y falsos positivos) en el umbral supuesto de una etiqueta individual. A medida que aumenta el umbral, es posible que el modelo haga menos predicciones. Sin embargo, en general, tendrá una proporción más alta de verdaderos positivos sobre falsos positivos en comparación con un umbral más bajo. Los valores posibles de precisión oscilan entre 0 y 1 y los valores más altos indican una precisión más alta.

Por ejemplo, cuando el modelo predice la presencia de un balón de fútbol en una imagen, ¿con qué frecuencia es correcta esa predicción? Supongamos que hay una imagen con 8 balones de fútbol y 5 rocas. Si el modelo predice 9 balones de fútbol (8 pronosticados correctamente y 1 falso positivo), la precisión de este ejemplo es de 0,89. Sin embargo, si el modelo ha predicho 13 balones de fútbol en la imagen con 8 predicciones correctas y 5 incorrectas, la precisión resultante es menor.

Para obtener más información, consulte Precisión y exhaustividad.

Exhaustividad

Etiquetas personalizadas de Amazon Rekognition genera métricas de exhaustividad promedio para cada etiqueta y una métrica de exhaustividad media para todo el conjunto de datos de prueba.

La exhaustividad es la fracción de las etiquetas del conjunto de pruebas que se ha predicho correctamente por encima del umbral supuesto. Con esto se mide la frecuencia con la que el modelo puede predecir correctamente una etiqueta personalizada cuando está realmente presente en las imágenes del conjunto de prueba. El rango de exhaustividad es de 0 a 1. Los valores más altos indican una exhaustividad más alta.

Por ejemplo, si una imagen contiene 8 balones de fútbol, ¿cuántos de ellos se han detectado correctamente? En este ejemplo en el que una imagen presenta 8 balones de fútbol y 5 rocas, si el modelo detecta 5 balones de fútbol, el valor de exhaustividad es de 0,62. Si después de volver a entrenarlo, el nuevo modelo detecta 9 balones de fútbol, incluidos los 8 que estaban presentes en la imagen, el valor de exhaustividad es de 1,0.

Para obtener más información, consulte Precisión y exhaustividad.

F1

Etiquetas personalizadas de Amazon Rekognition utiliza la métrica de puntuación F1 para medir el rendimiento medio del modelo en cada etiqueta y el rendimiento medio del modelo en todo el conjunto de datos de prueba.

El rendimiento del modelo es una medida acumulada que tiene en cuenta tanto la precisión como la exhaustividad de todas las etiquetas. (por ejemplo, la puntuación F1 o la precisión media). La puntuación de rendimiento del modelo es un valor entre 0 y 1. Cuanto mayor sea el valor, mejor será el rendimiento del modelo tanto en términos de exhaustividad como de precisión. En concreto, el rendimiento del modelo para las tareas de clasificación suele medirse mediante la puntuación F1. Esa puntuación es la media armónica de las puntuaciones de precisión y exhaustividad en el umbral supuesto. Por ejemplo, en un modelo con una precisión de 0,9 y una exhaustividad de 1,0, la puntuación F1 es de 0,947.

Un valor alto en la puntuación F1 indica que el modelo está funcionando bien tanto en precisión como en exhaustividad. Si el modelo no funciona como debiera, por ejemplo, y presenta una baja precisión de 0,30 y una alta exhaustividad de 1,0, la puntuación F1 será de 0,46. Del mismo modo, si la precisión es alta (0,95) y la exhaustividad es baja (0,20), la puntuación F1 será de 0,33. En ambos casos, la puntuación F1 es baja y da a entender que hay problemas con el modelo.

Para obtener más información, consulte Puntuación F1.

Uso de las métricas

En un modelo determinado que haya entrenado y en función de su aplicación, puede establecer un equilibrio entre precisión y exhaustividad utilizando el parámetro de entrada `MinConfidence` en `DetectCustomLabels`. Con un valor `MinConfidence` más alto, por lo general se obtiene una mayor precisión (más predicciones correctas de balones de fútbol), pero una menor exhaustividad (se pierden más balones de fútbol reales). Con un valor `MinConfidence` más bajo, se consigue una mayor exhaustividad (se pronostican correctamente más balones de fútbol reales), pero se reduce la precisión (habrá más predicciones que sean erróneas). Para obtener más información, consulte Análisis de una imagen con un modelo entrenado.

Las métricas también le informan sobre lo que podría hacer para mejorar el rendimiento del modelo si fuera necesario. Para obtener más información, consulte Mejora de un modelo de Etiquetas personalizadas de Amazon Rekognition.

> **Note**
>
> `DetectCustomLabels` devuelve las predicciones que van de 0 a 100, que se corresponden con el rango de métrica de 0 a 1.

Acceso a las métricas de evaluación (consola)

Durante las pruebas, se evalúa el rendimiento del modelo en comparación con el conjunto de datos de prueba. Las etiquetas del conjunto de datos de prueba se consideran «datos reales», ya que representan lo que se ve en la imagen real. Durante la prueba, el modelo hace predicciones utilizando el conjunto de datos de prueba. Las etiquetas pronosticadas se comparan con las etiquetas con datos reales y los resultados aparecen en la página de evaluación de la consola.

En la consola de Etiquetas personalizadas de Amazon Rekognition se registran las métricas resumidas de todo el modelo y las métricas de cada etiqueta. Las métricas disponibles en la consola son la precisión, la exhaustividad, la puntuación de F1, la confianza y el umbral de confianza. Para obtener más información, consulte Mejora de un modelo de Etiquetas personalizadas de Amazon Rekognition.

Puede utilizar la consola para centrarse en cada una de las métricas. Por ejemplo, para analizar los problemas de precisión de una etiqueta, puede filtrar los resultados del entrenamiento por etiqueta y por resultados falsos positivos. Para obtener más información, consulte Métricas para evaluar su modelo.

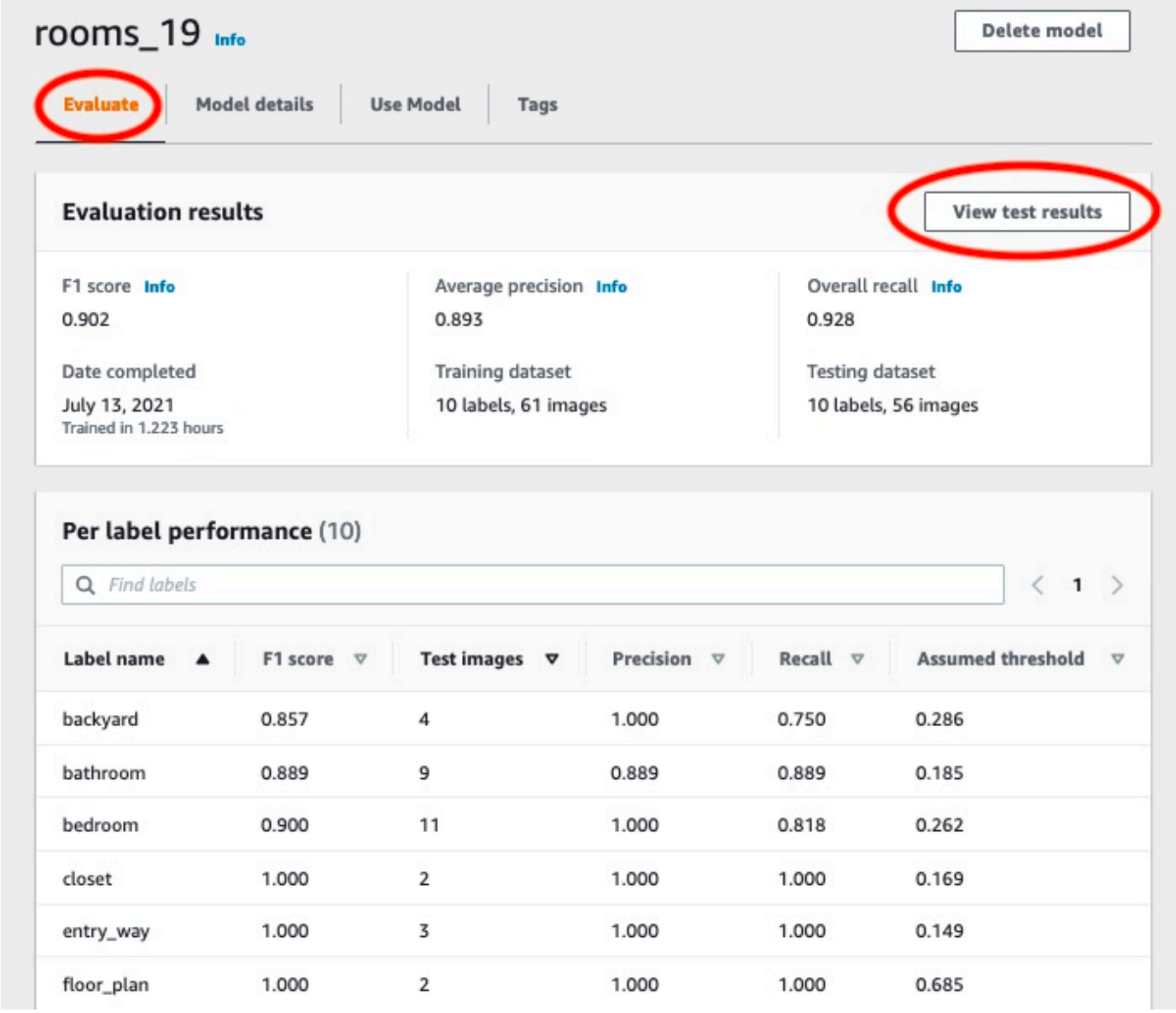
Tras el entrenamiento, el conjunto de datos de entrenamiento es de solo lectura. Si decide mejorar el modelo, puede copiar el conjunto de datos de entrenamiento en un nuevo conjunto de datos. Utilice la copia del conjunto de datos para entrenar una nueva versión del modelo.

En este paso, use la consola para acceder a los resultados de entrenamiento en ella.

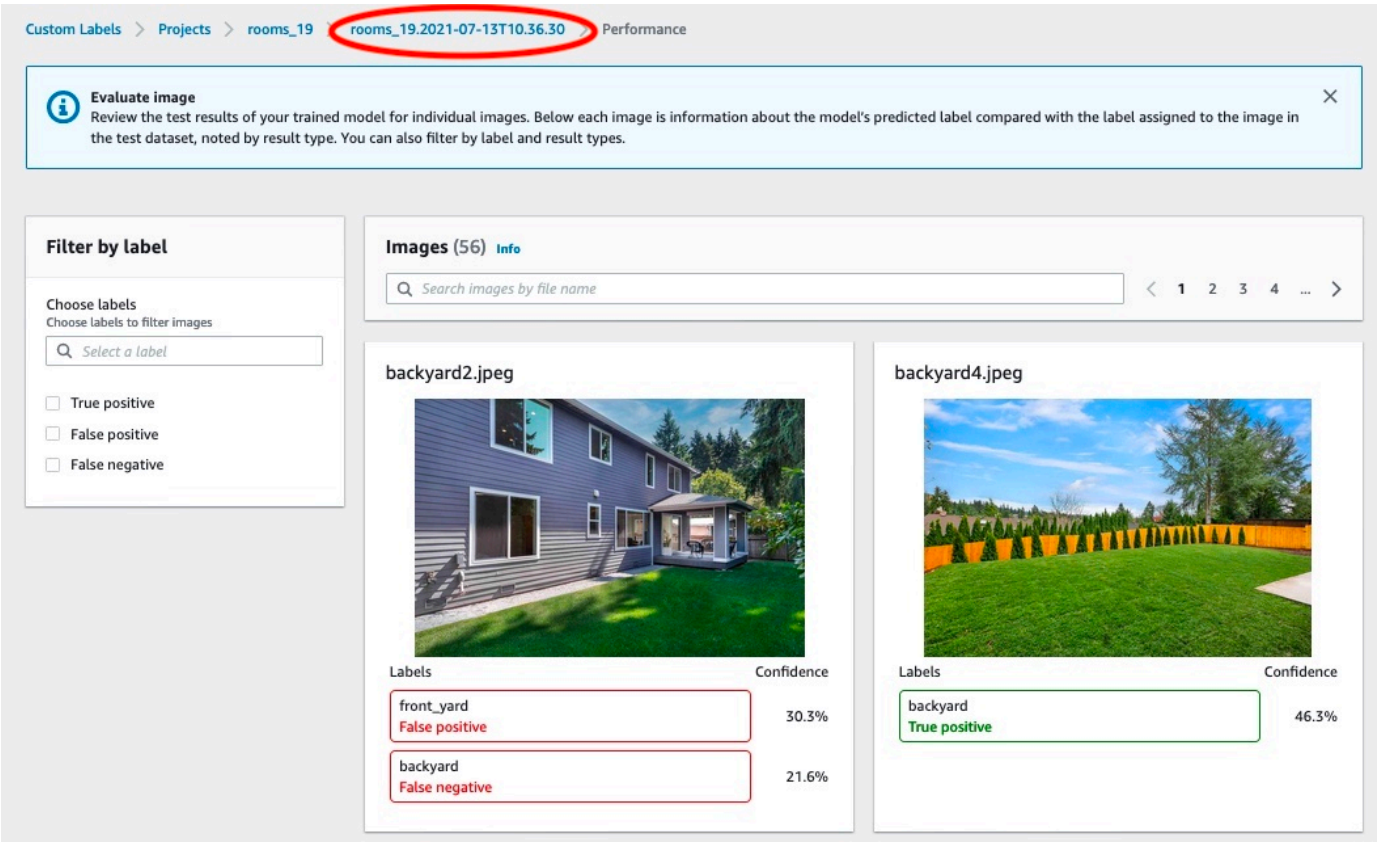
Cómo acceder a las métricas de evaluación (consola)

1. Abra la consola Amazon Rekognition en. https://console.aws.amazon.com/rekognition/
2. Elija Usar etiquetas personalizadas.
3. Elija Comenzar.
4. En el panel de navegación izquierdo, elija Proyectos.
5. En la página Proyectos, elija el proyecto que contiene el modelo entrenado que desea evaluar.
6. En Modelos, elija el modelo que desee evaluar.
7. Pulse el botón Evaluación para ver los resultados de la evaluación. Para obtener información sobre la evaluación de un modelo, consulte Mejora de un modelo de Etiquetas personalizadas de Amazon Rekognition.
8. Seleccione Ver resultados de pruebas para ver los resultados de cada una de las imágenes de prueba. Para obtener más información, consulte Métricas para evaluar su modelo. La siguiente

captura de pantalla del resumen de la evaluación del modelo muestra la puntuación F1, la precisión media y la recuperación global de 6 etiquetas con los resultados de las pruebas y las métricas de rendimiento. También se proporcionan detalles sobre el uso del modelo entrenado.



9. Tras ver los resultados de las pruebas, elija el nombre del proyecto para volver a la página del modelo. La página de resultados de la pruebas muestra imágenes con etiquetas pronosticadas y puntuaciones de confianza para un modelo de machine learning entrenado en las categorías de imagen de terraza y patio delantero. Se muestran dos imágenes de ejemplo.



10. Utilice las métricas para evaluar el rendimiento del modelo. Para obtener más información, consulte Mejora de un modelo de Etiquetas personalizadas de Amazon Rekognition.

Acceso a las métricas de evaluación de Etiquetas personalizadas de Amazon Rekognition (SDK)

La DescribeProjectVersionsoperación proporciona acceso a métricas más allá de las que se proporcionan en la consola.

Al igual que en la consola, `DescribeProjectVersions` da acceso a las siguientes métricas a modo de información resumida de los resultados de las pruebas y como resultados de las pruebas de cada etiqueta:

- Precisión
- Exhaustividad
- F1

Se devuelve el umbral medio de todas las etiquetas y el umbral de las etiquetas por separado.

`DescribeProjectVersions` también da acceso a las siguientes métricas para la clasificación y la detección de imágenes (ubicación de objetos en la imagen).

- Matriz de confusión para la clasificación de imágenes. Para obtener más información, consulte Visualización de la matriz de confusión en un modelo.
- Precisión promedio (MAP) para la detección de imágenes.
- Exhaustividad promedio (mAR) para la detección de imágenes.

`DescribeProjectVersions` también da acceso a valores verdaderos positivos, falsos positivos, falsos negativos y verdaderos negativos. Para obtener más información, consulte Métricas para evaluar su modelo.

La métrica acumulada de puntuación F1 la devuelve directamente `DescribeProjectVersions`. Se puede acceder a otras métricas a partir de una Acceder al archivo de resumen del modelo y de los archivos de Interpretación de la instantánea del manifiesto de evaluación almacenados en un bucket de Amazon S3. Para obtener más información, consulte Acceso al archivo de resumen y al resumen del manifiesto de evaluación (SDK).

Temas

- Acceder al archivo de resumen del modelo
- Interpretación de la instantánea del manifiesto de evaluación
- Acceso al archivo de resumen y al resumen del manifiesto de evaluación (SDK)
- Visualización de la matriz de confusión en un modelo
- Reference: archivo de resumen de los resultados del entrenamiento

Acceder al archivo de resumen del modelo

El archivo de resumen recoge los resultados de la evaluación, la información sobre el modelo en su conjunto y las métricas de cada etiqueta. Las métricas son precisión, exhaustividad y puntuación F1. También se incluye el valor umbral del modelo. Se puede acceder a la ubicación del archivo de resumen en el objeto `EvaluationResult` devuelto por `DescribeProjectVersions`. Para obtener más información, consulte Reference: archivo de resumen de los resultados del entrenamiento.

Consulte el siguiente ejemplo de un archivo de resumen.

```
{
  "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
    }
  ]
}
```

Interpretación de la instantánea del manifiesto de evaluación

El resumen de manifiesto de evaluación incluye información detallada sobre los resultados de la prueba. Este resumen registra el índice de confianza de cada predicción. También incluye la clasificación de la predicción en comparación con la clasificación real de la imagen (verdadero positivo, verdadero negativo, falso positivo o falso negativo).

Los archivos vienen resumidos, ya que solo se incluyen imágenes que podrían usarse para las pruebas y el entrenamiento. Las imágenes que no se pueden verificar, como las que tienen un formato incorrecto, no vienen incluidas en el manifiesto. Se puede acceder a la ubicación del resumen de las pruebas en el objeto `TestingDataResult` devuelto por `DescribeProjectVersions`. Se puede acceder a la ubicación del resumen del entrenamiento en el objeto `TrainingDataResult` devuelto por `DescribeProjectVersions`.

La instantánea está en el formato de salida del manifiesto de SageMaker AI Ground Truth y se añaden campos para proporcionar información adicional, como el resultado de la clasificación binaria de una detección. En el siguiente fragmento se ven los campos adicionales.

```
"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: la versión del formato del campo `rekognition-custom-labels-evaluation-details` incluida en el resumen del manifiesto.
- is-true-positive... — La clasificación binaria de la predicción en función de cómo se compara la puntuación de confianza con el umbral mínimo de la etiqueta.
- is-present-in-ground-verdad: Verdadero si la predicción realizada por el modelo está presente en la información de verdad básica utilizada en el entrenamiento; de lo contrario, es falsa. Este valor no se basa en si la puntuación de confianza supera el umbral mínimo calculado por el modelo.
- ground-truth-labeling-jobs— Una lista de los campos de verdad básica de la línea del manifiesto que se utilizan para el entrenamiento.

Para obtener información sobre el formato de manifiesto de SageMaker AI Ground Truth, consulta Resultados.

El siguiente es un ejemplo de un resumen del manifiesto de prueba donde figuran las métricas de la clasificación de imágenes y la detección de objetos.

```
// 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"
  }
}
```

Acceso al archivo de resumen y al resumen del manifiesto de evaluación (SDK)

Para obtener los resultados de la capacitación, llama DescribeProjectVersions. Para ver el código de ejemplo, consulte Descripción de un modelo (SDK).

La ubicación de las métricas se devuelve en la respuesta `ProjectVersionDescription` de `DescribeProjectVersions`.

- `EvaluationResult`: la ubicación del archivo de resumen.
- `TestingDataResult`: la ubicación del resumen del manifiesto de evaluación utilizada para las pruebas.

La puntuación F1 y la ubicación del archivo de resumen se devuelven en `EvaluationResult`. Por ejemplo:

```
"EvaluationResult": {
                "F1Score": 1.0,
                "Summary": {
                    "S3Object": {
                        "Bucket": "echo-dot-scans",
                        "Name": "test-output/EvaluationResultSummary-my-echo-dots-
project-v2.json"
                    }
                }
```

```
            }
```

El resumen del manifiesto de evaluación se almacena en la ubicación definida en el parámetro de entrada `--output-config` indicado en Entrenamiento de un modelo (SDK).

> **Note**
>
> Se devuelve en `BillableTrainingTimeInSeconds` la cantidad de tiempo, en segundos, que se le factura por el entrenamiento.

Para obtener información sobre las métricas que devuelve Etiquetas personalizadas de Amazon Rekognition, consulte Acceso a las métricas de evaluación de Etiquetas personalizadas de Amazon Rekognition (SDK) .

Visualización de la matriz de confusión en un modelo

Una matriz de confusión le permite ver las etiquetas que el modelo confunde con otras etiquetas del modelo. Al utilizar una matriz de confusión, puede ajustar más el modelo y mejorarlo.

Durante la evaluación del modelo, Etiquetas personalizadas de Amazon Rekognition crea una matriz de confusión al utilizar las imágenes de prueba para identificar etiquetas mal identificadas (confusas). Etiquetas personalizadas de Amazon Rekognition solo crea una matriz de confusión para los modelos de clasificación. Se puede acceder a la matriz de clasificación en el archivo de resumen que Etiquetas personalizadas de Amazon Rekognition crea durante el entrenamiento del modelo. No es posible ver la matriz de confusión en la consola de Etiquetas personalizadas de Amazon Rekognition.

Temas

- Uso de una matriz de confusión
- Cómo obtener la matriz de confusión en un modelo

Uso de una matriz de confusión

La siguiente tabla es la matriz de confusión del proyecto de ejemplo con clasificación de imágenes de habitaciones. Los encabezados de las columnas son las etiquetas (etiquetas con datos reales) asignadas a las imágenes de prueba. Los encabezados de fila son las etiquetas que el modelo

predice para las imágenes de prueba. Cada celda es el porcentaje de predicciones de una etiqueta (fila) que debe ser la etiqueta de datos reales (columna). Por ejemplo, el 67 % de las predicciones para los baños se etiquetaron correctamente como baños. El 33 % de los baños se etiquetaron incorrectamente como cocinas. Un modelo de alto rendimiento tiene valores de celda altos cuando la etiqueta pronosticada coincide con la etiqueta de datos reales. Puede verlas como una línea diagonal desde la primera hasta la última etiqueta pronosticada y las etiquetas de datos reales. Si el valor de una celda es 0, no se ha realizado ninguna predicción en la etiqueta de predicción de la celda, que debe ser la etiqueta de datos reales de la celda.

> **Note**
>
> Como los modelos no son deterministas, los valores de las celdas de la matriz de confusión que se obtienen al entrenar el proyecto Habitaciones pueden variar de los de la siguiente tabla.

La matriz de confusión identifica las áreas en las que centrarse. Por ejemplo, la matriz de confusión refleja que el 50 % de las veces el modelo confundió armarios con dormitorios. En esta situación, se deben agregar más imágenes de armarios y dormitorios al conjunto de datos de entrenamiento. Compruebe también que las imágenes de armarios y dormitorios existentes estén bien etiquetadas. Esto debería ayudar al modelo a distinguir mejor entre las dos etiquetas. Para agregar más imágenes a un conjunto de datos, consulte Agregar más imágenes a un conjunto de datos.

Aunque la matriz de confusión es útil, es importante tener en cuenta otras métricas. Por ejemplo, el 100 % de las predicciones detectaron correctamente la etiqueta plano_planta, lo que indica un rendimiento excelente. Sin embargo, el conjunto de datos de prueba solo tiene 2 imágenes con la etiqueta plano_planta. También hay 11 imágenes con la etiqueta sala_estar. Este desajuste también se encuentra en el conjunto de datos de entrenamiento (13 imágenes con sala_estar y 2 imágenes de armarios). Para lograr una evaluación más precisa, equilibre los conjuntos de datos de entrenamiento y de prueba añadiendo más imágenes de etiquetas infrarrepresentadas (planos de planta en este ejemplo). Para obtener el número de imágenes de prueba por etiqueta, consulte Acceso a las métricas de evaluación (consola).

La siguiente tabla es un ejemplo de matriz de confusión en la que se compara la etiqueta pronosticada (en el eje y) con la etiqueta de datos reales:

Etiqueta pronosticada	terraza	baño	dormitorio	armario	camino_entrada	plano_planta	patio_delantero	cocina	sala_estar	patio
terraza	75%	0%	0%	0%	0%	0%	33%	0%	0%	0%
baño	0%	67%	0%	0%	0%	0%	0%	0%	0%	0%
dormitorio	0%	0%	82%	50%	0%	0%	0%	0%	9%	0%
armario	0%	0%	0%	50%	0%	0%	0%	0%	0%	0%
camino_entrada	0%	0%	0%	0%	33%	0%	0%	0%	0%	0%
plano_planta	0%	0%	0%	0%	0%	100%	0%	0%	0%	0%
patio_delantero	25%	0%	0%	0%	0%	0%	67%	0%	0%	0%
cocina	0%	33%	0%	0%	0%	0%	0%	88%	0%	0%
sala_estar	0%	0%	18%	0%	67%	0%	0%	12%	91%	33%
patio	0%	0%	0%	0%	0%	0%	0%	0%	0%	67%

Cómo obtener la matriz de confusión en un modelo

El código siguiente utiliza las DescribeProjectVersionsoperaciones DescribeProjectsy para obtener el archivo de resumen de un modelo. Tras esto, este utiliza el archivo de resumen para mostrar la matriz de confusión del modelo.

Cómo ver la matriz de confusión de un modelo (SDK)

1. Si aún no lo ha hecho, instale y configure las AWS CLI y las AWS SDKs. Para obtener más información, consulte Paso 4: Configure y AWS CLIAWS SDKs.

2. Utilice el siguiente código para ver la matriz de confusión de un modelo. Indique los siguientes argumentos de línea de comandos:

   - `project_name`: el nombre del proyecto que desea usar. Puede obtener el nombre del proyecto en la página de proyectos en la consola de Etiquetas personalizadas de Amazon Rekognition.
   - `version_name`: la versión del modelo que desea utilizar. Puede obtener el nombre de la versión en la página de detalles del proyecto en la consola de Etiquetas personalizadas de Amazon Rekognition.

```
# 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()
```

Reference: archivo de resumen de los resultados del entrenamiento

El resumen de los resultados del entrenamiento incluye métricas que puede utilizar para evaluar el modelo. El archivo de resumen también sirve para ver las métricas en la página de resultados del entrenamiento en la consola. El archivo de resumen se almacena en un bucket de Amazon S3 tras el entrenamiento. Para obtener el archivo de resumen, llame a `DescribeProjectVersion`. Para ver el código de ejemplo, consulte Acceso al archivo de resumen y al resumen del manifiesto de evaluación (SDK).

Archivo de resumen

El siguiente JSON es el formato del archivo de resumen.

EvaluationDetails (sección 3)

Información general sobre la tarea de entrenamiento. Esto incluye el ARN del proyecto al que pertenece el modelo (`ProjectVersionArn`)), la fecha y hora en que terminó el entrenamiento, la versión del modelo que se ha evaluado (`EvaluationEndTimestamp`) y una lista de etiquetas detectadas durante el entrenamiento (`Labels`). También se registra el número de imágenes utilizadas para el entrenamiento (`NumberOfTrainingImages`) y la evaluación (`NumberOfTestingImages`).

AggregatedEvaluationResults (sección 1)

Se puede utilizar `AggregatedEvaluationResults` para evaluar el rendimiento general del modelo entrenado cuando se utiliza con el conjunto de datos de prueba. Se incluyen métricas acumuladas para las métricas `Precision`, `Recall` y `F1Score`. Para la detección de objetos (la ubicación de objetos en una imagen), se devuelven las métricas `AverageRecall` (mAR) y `AveragePrecision` (MaP). Para la clasificación (el tipo de objeto en una imagen), se devuelve una métrica de matriz de confusión.

LabelEvaluationResults (sección 2)

Se puede utilizar `labelEvaluationResults` para evaluar el rendimiento de etiquetas concretas. Las etiquetas se ordenan según la puntuación F1 de cada etiqueta. Las métricas incluidas son `Precision`, `Recall`, `F1Score` y `Threshold` (utilizadas para la clasificación).

El nombre de archivo tiene el formato siguiente: `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"
      },
    },
    ...
  ]
}
```

Mejora de un modelo de Etiquetas personalizadas de Amazon Rekognition

El rendimiento de los modelos de machine learning depende en gran medida de factores como la complejidad y la variabilidad de las etiquetas personalizadas (los objetos y escenas específicos que le interesen), la calidad y la capacidad representativa del conjunto de datos de entrenamiento que usa, los esquemas de modelos y los métodos de machine learning para entrenar el modelo.

Etiquetas personalizadas de Amazon Rekognition simplifica este proceso y no se requiere conocimientos de aprendizaje automático. Sin embargo, el proceso de creación de un buen modelo

a menudo implica iteraciones sobre los datos y mejoras en el modelo para lograr el rendimiento deseado. A continuación, encontrará información sobre cómo mejorar su modelo.

Datos

En general, se puede mejorar la calidad del modelo con más cantidades de datos y de mejor calidad. Utilice imágenes de entrenamiento que muestren claramente el objeto o la escena y que no estén repletas de elementos innecesarios. Para delimitar los cuadros alrededor de los objetos, use imágenes de entrenamiento que reflejen el objeto de forma totalmente visible y no se vea obstaculizado por otros objetos.

Asegúrese de que los conjuntos de datos de entrenamiento y de prueba coincidan con el tipo de imágenes con las que vaya a realizar la inferencia. En el caso de los objetos, como los logotipos, en los que solo hay algunos ejemplos de entrenamiento, debes incluir cuadros delimitadores alrededor del logotipo en las imágenes de prueba. Estas imágenes representan o ilustran los escenarios en los que desea localizar el objeto.

Para agregar más imágenes a un conjunto de datos de entrenamiento o de prueba, consulte Agregar más imágenes a un conjunto de datos.

Reducción de los falsos positivos (mayor precisión)

- En primer lugar, compruebe si el hecho de aumentar el umbral supuesto permite que las predicciones sean correctas y, al mismo tiempo, reducir los falsos positivos. En algún momento, esto merma los avances debido a la compensación entre precisión y exhaustividad en un modelo determinado. No se puede establecer el umbral supuesto de una etiqueta, pero se puede lograr el mismo resultado si se indica un valor alto en el parámetro de entrada `MinConfidence` en `DetectCustomLabels`. Para obtener más información, consulte Análisis de una imagen con un modelo entrenado.
- Es posible que observe que una o varias de las etiquetas personalizadas que le interesan (A) se confunden constantemente con la misma clase de objetos (pero no con una etiqueta que le interese) (B). Para ayudarte, agregue B como etiqueta de clase de objeto al conjunto de datos de entrenamiento (junto con las imágenes en las que haya recibido el falso positivo). De hecho, así estará ayudando al modelo a saber predecir B y no A gracias a las nuevas imágenes de entrenamiento. Para agregar imágenes a un conjunto de datos de entrenamiento, consulte Agregar más imágenes a un conjunto de datos.
- Es posible que dos de las etiquetas personalizadas (A y B) hagan confundir al modelo; se pronostica que la imagen de prueba con la etiqueta A tendrá la etiqueta B y viceversa. En ese

caso, compruebe primero si hay imágenes mal etiquetadas en los conjuntos de entrenamiento y de prueba. Use la galería de conjuntos de datos para administrar las etiquetas asignadas a un conjunto de datos. Para obtener más información, consulte Administración de etiquetas. Además, si agrega más imágenes de entrenamiento relacionadas con este tipo de confusión hará que un modelo reentrenado discrimine mejor entre A y B. Para agregar imágenes a un conjunto de datos de entrenamiento, consulte Agregar más imágenes a un conjunto de datos.

Reducción de los falsos negativos (mejor exhaustividad)

- Indique un valor inferior para el umbral supuesto. No se puede establecer el umbral supuesto de una etiqueta, pero se puede lograr el mismo resultado si se indica un valor bajo en el parámetro de entrada `MinConfidence` en `DetectCustomLabels`. Para obtener más información, consulte Análisis de una imagen con un modelo entrenado.
- Use mejores ejemplos para que haya más variedad tanto del objeto como de las imágenes en las que aparecen.
- Divida la etiqueta en dos clases que sean más fáciles de aprender. Por ejemplo, en lugar de galletas buenas y malas, tal vez prefiera galletas buenas, galletas quemadas y galletas en pedacitos para que el modelo pueda aprender mejor cada concepto único.

Ejecución de un modelo de Etiquetas personalizadas de Amazon Rekognition

Si el modelo ya tiene la precisión que busca, puede empezar a utilizarlo. Puede iniciar y detener un modelo mediante la consola o el AWS SDK. La consola también incluye ejemplos de operaciones del SDK que puede utilizar.

Temas

Unidades de inferencia

Al iniciar el modelo, se indica el número de recursos de computación, conocidos como unidades de inferencia, que utiliza el modelo.

> **Important**
>
> Se le cobrará por el número de horas que el modelo esté en ejecución y por el número de unidades de inferencia que utilice mientras se ejecute, en función de cómo configure la ejecución del modelo. Por ejemplo, si inicia el modelo con dos unidades de inferencia y lo utiliza durante 8 horas, se le cobrarán 16 horas de inferencia (8 horas de tiempo de ejecución x dos unidades de inferencia). Para obtener más información, consulte Horas de inferencias. Si no detiene el modelo directamente, se le cobrará aunque no se analicen activamente las imágenes con el modelo.

Las transacciones por segundo (TPS) que engloba una sola unidad de inferencia se ven afectadas por lo siguiente.

- Un modelo que detecta etiquetas de imagen (clasificación) generalmente tiene un TPS más alto que un modelo que detecta y localiza objetos con cuadros delimitadores (detección de objetos).

- La complejidad del modelo.
- Una imagen con mayor resolución exige más tiempo de análisis.
- El análisis de más objetos en una imagen requiere más tiempo.
- Las imágenes más pequeñas se analizan más rápido que las imágenes más grandes.
- Una imagen transferida como bytes de imagen se analiza más rápido que al cargar primero la imagen en un bucket de Amazon S3 y, a continuación, habilitar un enlace a la imagen cargada. Las imágenes transferidas como bytes de imagen deben ocupar menos de 4 MB. Se recomienda utilizar los bytes de imagen para el procesamiento de imágenes prácticamente en tiempo real y cuando el tamaño de la imagen sea inferior a 4 MB. Por ejemplo, las imágenes capturadas con una cámara de vigilancia.
- Si se procesan las imágenes almacenadas en un bucket de Amazon S3, el proceso será más rápido que descargarlas, convertirlas en bytes de imagen y luego transferirlas para su análisis.
- Si se analiza una imagen ya almacenada en un bucket de Amazon S3, probablemente el proceso sea más rápido que analizar la misma imagen transferida como bytes de imagen. Esto es especialmente cierto si el tamaño de la imagen es superior.

Si el número de llamadas a `DetectCustomLabels` supera el TPS máximo admitido por el total de unidades de inferencia que utiliza un modelo, Etiquetas personalizadas de Amazon Rekognition devolverá la excepción `ProvisionedThroughputExceededException`.

Gestión del rendimiento con unidades de inferencia

Puede aumentar o disminuir el rendimiento del modelo en función de las exigencias de su aplicación. Para aumentar el rendimiento, utilice unidades de inferencia adicionales. Cada unidad de inferencia adicional aumenta la velocidad de procesamiento en una unidad de inferencia. Para obtener información sobre cómo calcular el número de unidades de inferencia que necesita, consulte Calcular unidades de inferencia en los modelos Etiquetas personalizadas de Amazon Rekognition y Amazon Lookout for Vision. Si desea cambiar el rendimiento permitido para su modelo, tiene dos opciones:

Cómo añadir o eliminar unidades de inferencia manualmente

Detenga el modelo y luego reinícielo con el número correspondiente de unidades de inferencia. La desventaja de este método es que el modelo no puede recibir solicitudes mientras se reinicia y no se puede utilizar para gestionar los picos de demanda. Utilice este método si su modelo tiene un

rendimiento constante y en su caso podría tolerar entre 10 y 20 minutos de inactividad. Un ejemplo de ello sería si deseara realizar llamadas por lotes en su modelo utilizando un calendario semanal.

Escalado automático de unidades de inferencia

Si su modelo tiene que adaptarse a picos de demanda, Etiquetas personalizadas de Amazon Rekognition ahora puede escalar el número de unidades de inferencia que utiliza el modelo. A medida que aumenta la demanda, Etiquetas personalizadas de Amazon Rekognition agrega unidades de inferencia adicionales al modelo y las elimina cuando la demanda decrece.

Para permitir que Etiquetas personalizadas de Amazon Rekognition escale automáticamente las unidades de inferencia de un modelo, inícielo y elija el número máximo de unidades de inferencia que puede utilizar mediante el parámetro `MaxInferenceUnits`. Si define un número máximo de unidades de inferencia, podrá administrar los costes de ejecutar el modelo al limitar el número de unidades de inferencia disponibles. Si no indica un número máximo de unidades, Etiquetas personalizadas de Amazon Rekognition no escalará automáticamente el modelo, sino que solo utilizará el número de unidades de inferencia con las que comenzó. Para obtener información sobre el número máximo de unidades de inferencia, consulte Service Quotas.

También puede indicar un número mínimo de unidades de inferencia mediante el parámetro `MinInferenceUnits`. Esto le permite especificar el rendimiento mínimo del modelo, donde una sola unidad de inferencia representa 1 hora de tiempo de procesamiento.

> **Note**
>
> No puede definir el número máximo de unidades de inferencia en la consola de Etiquetas personalizadas de Amazon Rekognition. En su lugar, indique el parámetro de entrada `MaxInferenceUnits` en la operación `StartProjectVersion`.

Amazon Rekognition Custom Labels proporciona las siguientes métricas de CloudWatch Amazon Logs que puede utilizar para determinar el estado actual de escalado automático de un modelo.

Métrica	Descripción
`DesiredInferenceUnits`	El número de unidades de inferencia con las que Etiquetas personalizadas de Amazon Rekognition está haciendo la escala más grande o pequeña.

Métrica	Descripción
`InServiceInferenceUnits`	El número de unidades de inferencia que utiliza el modelo.

Si `DesiredInferenceUnits` = `InServiceInferenceUnits`, Etiquetas personalizadas de Amazon Rekognition no estará escalando el número de unidades de inferencia.

Si `DesiredInferenceUnits` > `InServiceInferenceUnits`, Etiquetas personalizadas de Amazon Rekognition aumentará la escala más grande hasta alcanzar el valor de `DesiredInferenceUnits`.

Si `DesiredInferenceUnits` > `InServiceInferenceUnits`, Etiquetas personalizadas de Amazon Rekognition reducirá la escala más pequeña hasta alcanzar el valor de `DesiredInferenceUnits`.

Para obtener más información sobre las métricas devueltas por las etiquetas personalizadas de Amazon Rekognition y las dimensiones de filtrado, CloudWatch consulte las métricas de Rekognition.

Para saber el número máximo de unidades de inferencia que ha solicitado para un modelo, llame a `DescribeProjectsVersion` y revise el campo `MaxInferenceUnits` en la respuesta. Para ver el código de ejemplo, consulte Descripción de un modelo (SDK).

Zonas de disponibilidad

Etiquetas personalizadas de Amazon Rekognition distribuye las unidades de inferencia en varias zonas de disponibilidad de una región de AWS para ofrecer una mayor disponibilidad. Para obtener más información, consulte Zonas de disponibilidad. Para proteger los modelos de producción de las interrupciones en las zonas de disponibilidad y de los errores en las unidades de inferencia, inicie los modelos de producción con al menos dos unidades de inferencia.

Si se produce una interrupción en la zona de disponibilidad, todas las unidades de inferencia de la zona de disponibilidad no estarán disponibles y la capacidad del modelo se reducirá. Las llamadas a se redistribuyen entre las unidades de DetectCustomLabelsinferencia restantes. Estas llamadas se realizan correctamente si no superan las transacciones por segundo (TPS) permitidas de las unidades de inferencia restantes. Una vez que AWS repara la zona de disponibilidad, se reinician las unidades de inferencia y se restablece la capacidad total.

Si una sola unidad de inferencia falla, Etiquetas personalizadas de Amazon Rekognition iniciará automáticamente una nueva unidad de inferencia en la misma zona de disponibilidad. La capacidad del modelo se reduce hasta que se inicie la nueva unidad de inferencia.

Inicio de un modelo de Etiquetas personalizadas de Amazon Rekognition

Puede empezar a ejecutar un modelo de Amazon Rekognition Custom Labels mediante la consola o mediante la operación. StartProjectVersion

> **Important**
>
> Se le cobrará por el número de horas que el modelo esté en ejecución y por el número de unidades de inferencia que utilice mientras se ejecute. Para obtener más información, consulte Ejecución de un modelo de Etiquetas personalizadas de Amazon Rekognition.

El proceso de inicio de un modelo puede tardar algunos minutos. Para comprobar el estado actual de preparación del modelo, consulte la página de detalles del proyecto o uso. DescribeProjectVersions

Una vez iniciado el modelo DetectCustomLabels, utilice, para analizar las imágenes utilizando el modelo. Para obtener más información, consulte Análisis de una imagen con un modelo entrenado. La consola también incluye un código de ejemplo para llamar a `DetectCustomLabels`.

Temas

- Inicio de un modelo de Etiquetas personalizadas de Amazon Rekognition (consola)
- Inicio de un modelo de Etiquetas personalizadas de Amazon Rekognition (SDK)

Inicio de un modelo de Etiquetas personalizadas de Amazon Rekognition (consola)

Utilice el siguiente procedimiento para empezar a ejecutar un modelo de Etiquetas personalizadas de Amazon Rekognition con la consola. Puede iniciar el modelo directamente desde la consola o utilizar el código del AWS SDK proporcionado por la consola.

Cómo iniciar un modelo (consola)

1. Abra la consola Amazon Rekognition en. https://console.aws.amazon.com/rekognition/
2. Elija Usar etiquetas personalizadas.
3. Elija Comenzar.
4. En el panel de navegación izquierdo, elija Proyectos.
5. En la página Proyectos, elija el proyecto que contiene el modelo entrenado que quiera iniciar.
6. En la sección Modelos, elija el modelo que desee iniciar.
7. Seleccione la pestaña Usar modelo.
8. Realice una de las siguientes acciones:

   Start model using the console

   En la sección Iniciar o detener modelo, haga lo siguiente:

   1. Seleccione el número de unidades de inferencia que desee utilizar. Para obtener más información, consulte Ejecución de un modelo de Etiquetas personalizadas de Amazon Rekognition.
   2. Elija Iniciar.
   3. En el cuadro de diálogo Iniciar modelo, seleccione Iniciar.

   Start model using the AWS SDK

   En la sección Usar su modelo, realice lo siguiente:

   1. Elija Código de API.
   2. Elija AWS CLI o Python.
   3. En Iniciar modelo, copie el código de ejemplo.
   4. Utilice el código de ejemplo para iniciar el modelo. Para obtener más información, consulte Inicio de un modelo de Etiquetas personalizadas de Amazon Rekognition (SDK).
9. Para volver a la página de información general del proyecto, elija el nombre del proyecto en la parte superior de la página.
10. En la sección Modelo, compruebe el estado del modelo. Cuando el estado del modelo esté EN EJECUCIÓN, puede utilizar el modelo para analizar imágenes. Para obtener más información, consulte Análisis de una imagen con un modelo entrenado.

Inicio de un modelo de Etiquetas personalizadas de Amazon Rekognition (SDK)

Para iniciar un modelo, se llama a la StartProjectVersionAPI y se pasa el nombre de recurso de Amazon (ARN) del modelo en el parámetro de `ProjectVersionArn` entrada. Indique también el número de unidades de inferencia que desee utilizar. Para obtener más información, consulte Ejecución de un modelo de Etiquetas personalizadas de Amazon Rekognition.

Es posible que un modelo tarde un tiempo en iniciarse. En los ejemplos de Python y Java de este tema, se utilizan esperadores para que se inicie el modelo. Un esperador es un método de utilidad que sondean si se da un determinado estado. Como alternativa, puede comprobar el estado actual llamando DescribeProjectVersions.

Cómo iniciar un modelo (SDK)

1. Si aún no lo ha hecho, instale y configure el AWS CLI y el AWS SDKs. Para obtener más información, consulte Paso 4: Configure y AWS CLIAWS SDKs.
2. Utilice el código de ejemplo para iniciar un modelo.

   CLI

   Cambie el valor de `project-version-arn` por el ARN del modelo que desee iniciar. Cambie el valor de `--min-inference-units` por el número de unidades de inferencia que desee utilizar. Si lo desea, cambie `--max-inference-units` por el número máximo de unidades de inferencia que Etiquetas personalizadas de Amazon Rekognition puede utilizar para escalar automáticamente el modelo.

   ```
   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

   Indique los siguientes parámetros de línea de comandos:

   - `project_arn`: el ARN del proyecto que incluye el modelo que desea iniciar.
   - `model_arn`: el ARN del modelo que desea iniciar.

- `min_inference_units`: el número de unidades de inferencia que desea utilizar.
- (Opcional) `--max_inference_units`: el número máximo de unidades de inferencia que Etiquetas personalizadas de Amazon Rekognition puede utilizar para escalar automáticamente el modelo.

```
# 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

Indique los siguientes parámetros de línea de comandos:

- `project_arn`: el ARN del proyecto que incluye el modelo que desea iniciar.
- `model_arn`: el ARN del modelo que desea iniciar.
- `min_inference_units`: el número de unidades de inferencia que desea utilizar.
- (Opcional)`max_inference_units`: el número máximo de unidades de inferencia que Etiquetas personalizadas de Amazon Rekognition puede utilizar para escalar automáticamente el modelo. Si no se indica ningún valor, no se aplicará el escalado automático.

```
/*
   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);
        }



    }

}
```

Detención de un modelo de Etiquetas personalizadas de Amazon Rekognition

Puede dejar de ejecutar un modelo de Amazon Rekognition Custom Labels mediante la consola o mediante la operación. StopProjectVersion

Temas

- Detención de un modelo de Etiquetas personalizadas de Amazon Rekognition (consola)
- Detención de un modelo de Etiquetas personalizadas de Amazon Rekognition (SDK)

Detención de un modelo de Etiquetas personalizadas de Amazon Rekognition (consola)

Utilice el siguiente procedimiento para dejar de ejecutar un modelo de Etiquetas personalizadas de Amazon Rekognition con la consola. Puede detener el modelo directamente desde la consola o utilizar el código del AWS SDK proporcionado por la consola.

Cómo detener un modelo (consola)

1. Abra la consola Amazon Rekognition en. https://console.aws.amazon.com/rekognition/
2. Elija Usar etiquetas personalizadas.
3. Elija Comenzar.
4. En el panel de navegación izquierdo, elija Proyectos.
5. En la página Proyectos, elija el proyecto que contiene el modelo entrenado que quiera detener.
6. En la sección Modelos, elija el modelo que desee detener.
7. Seleccione la pestaña Usar modelo.
8. Stop model using the console

    1. En la sección Iniciar o detener modelo, seleccione Detener.
    2. En el cuadro de diálogo Detener modelo, escriba detener para confirmar que desea detener el modelo.
    3. Seleccione Detener para detener el modelo.

    Stop model using the AWS SDK

    En la sección Usar su modelo, realice lo siguiente:

    1. Elija Código de API.
    2. Elija AWS CLI o Python.
    3. En Detener modelo, copie el código de ejemplo.
    4. Utilice el código de ejemplo para detener el modelo. Para obtener más información, consulte Detención de un modelo de Etiquetas personalizadas de Amazon Rekognition (SDK).
9. Elija el nombre del proyecto en la parte superior de la página para volver a la página de información general del proyecto.

10. En la sección Modelo, compruebe el estado del modelo. El modelo se habrá detenido cuando el estado del modelo esté en DETENIDO.

Detención de un modelo de Etiquetas personalizadas de Amazon Rekognition (SDK)

Para detener un modelo, se llama a la StopProjectVersionAPI y se pasa el nombre de recurso de Amazon (ARN) del modelo en el parámetro de `ProjectVersionArn` entrada.

Es posible que el modelo tarde un tiempo en detenerse. Para comprobar el estado actual, use `DescribeProjectVersions`.

Cómo detener un modelo (SDK)

1. Si aún no lo ha hecho, instale y configure el AWS CLI y el AWS SDKs. Para obtener más información, consulte Paso 4: Configure y AWS CLIAWS SDKs.
2. Utilice el código de ejemplo para iniciar un modelo.

   CLI

   Cambie el valor de `project-version-arn` por el ARN de la versión del modelo que desee detener.

   ```
   aws rekognition stop-project-version --project-version-arn "model arn" \
     --profile custom-labels-access
   ```

   Python

   En el siguiente ejemplo se detiene un modelo que ya se está ejecutando.

   Indique los siguientes parámetros de línea de comandos:

   - `project_arn`: el ARN del proyecto que incluye el modelo que desea detener.
   - `model_arn`: el ARN del modelo que desea detener.

   ```
   # 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

Indique los siguientes parámetros de línea de comandos:

- `project_arn`: el ARN del proyecto que incluye el modelo que desea detener.
- `model_arn`: el ARN del modelo que desea detener.

```
/*
   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);
        }

    }

}
```

Cómo obtener informes sobre la duración de la ejecución y las unidades de inferencia utilizadas

Si entrenó e inició su modelo después de agosto de 2022, puede usar la CloudWatch métrica de `InServiceInferenceUnits` Amazon para determinar cuántas horas ha durado un modelo y la cantidad de unidades de inferencia utilizadas durante esas horas.

> **Note**
>
> Si solo tiene un modelo en una AWS región, también puede obtener el tiempo de ejecución del modelo haciendo un seguimiento de las llamadas entrantes `StartprojectVersion` y `StopProjectVersion` entrantes que se hayan realizado correctamente. CloudWatch Este enfoque no funciona si se ejecuta más de un modelo en la AWS región, ya que las métricas no incluyen información sobre el modelo.
>
> Como alternativa, puede utilizarlo AWS CloudTrail para realizar un seguimiento de las llamadas a `StartProjectVersion` y `StopProjectVersion` (que incluye el ARN modelo en el `requestParameters` campo del historial de eventos). CloudTrail los eventos están limitados a 90 días, pero puedes almacenar eventos de hasta 7 años en un CloudTraillago.

Con el siguiente procedimiento se pueden crear gráficas para lo siguiente:

- El número de horas que ha estado en funcionamiento un modelo.
- El número de unidades de inferencia que ha utilizado el modelo.

Puede elegir un período de tiempo de hasta 15 meses atrás. Para obtener más información sobre la retención de métricas, consulte Retención de métricas.

Cómo determinar la duración del modelo y las unidades de inferencia utilizadas en un modelo

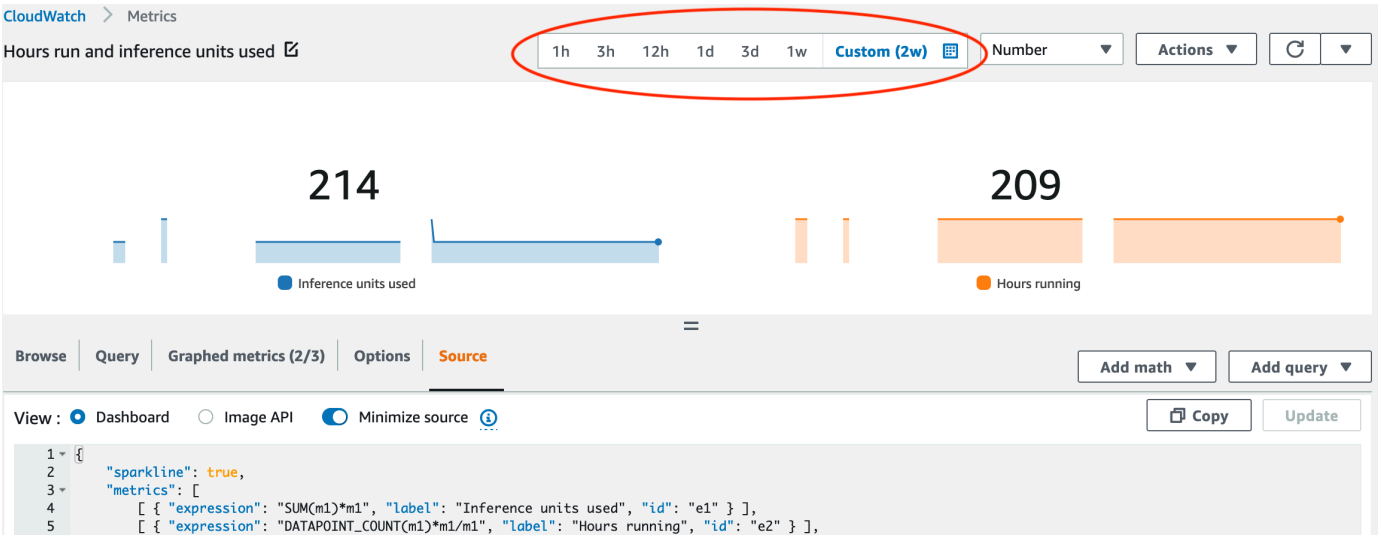
1. Inicie sesión en AWS Management Console y abra la CloudWatch consola en https://console.aws.amazon.com/cloudwatch/.
2. En el panel de navegación izquierdo, elija Todas las métricas en Métricas.
3. En el panel de contenido, elija la pestaña Origen.
4. Asegúrese de que el botón Panel esté seleccionado.
5. En el cuadro de edición, reemplace el JSON existente por el siguiente. Cambie los siguientes valores:

   - `Project_Name`: el proyecto que incluye el modelo del que desea hacer una gráfica.
   - `Version_Name`: la versión del modelo del que desea hacer una gráfica.
   - `AWS_Region`— La AWS región que contiene el modelo. Asegúrese de que la CloudWatch consola esté en la misma AWS región. Para ello, compruebe el selector de regiones de la barra de navegación de la parte superior de la página. Realice los cambios necesarios.

```
{
    "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. Elija Actualizar.
7. En la parte superior de la página, elija una línea de tiempo. Debería ver los números de las unidades de inferencia utilizadas y las horas de funcionamiento a lo largo de la línea de tiempo. Los vacíos en la gráfica indican los momentos en los que el modelo no estaba en ejecución. La siguiente captura de pantalla de la consola muestra las unidades de inferencia utilizadas y las horas acumuladas en períodos de tiempo, con un tiempo personalizado establecido de 2 semanas, con los valores más altos de 214 unidades de inferencia y 209 horas de ejecución.



8. (Opcional) Para agregar la gráfica a un panel, elija Acciones y después Agregar al panel.

Análisis de una imagen con un modelo entrenado

Para analizar una imagen con un modelo de Amazon Rekognition Custom Labels entrenado, debe llamar a la API. DetectCustomLabels El resultado de `DetectCustomLabels` es una predicción de que la imagen incluye objetos, escenas o conceptos específicos.

Para llamar a `DetectCustomLabels`, debe indicar lo siguiente:

- El nombre de recurso de Amazon (ARN) del modelo de Etiquetas personalizadas de Amazon Rekognition que desea utilizar.
- La imagen con la que desea que el modelo haga una predicción. Puede facilitar una imagen de entrada como matriz de bytes de imagen (bytes de imagen con codificación base64) o como objeto de Amazon S3. Para obtener más información, consulte Imagen.

Las etiquetas personalizadas se devuelven en una matriz de objetos de Etiqueta personalizada. Cada etiqueta personalizada representa un único objeto, escena o concepto que se encuentra en la imagen. Una etiqueta personalizada incluye:

- Una etiqueta del objeto, de la escena o del concepto que se encuentra en la imagen.
- Un cuadro delimitador de los objetos que se encuentran en la imagen. Con las coordenadas del cuadro delimitador se indica dónde se encuentra el texto en la imagen de origen. Los valores de las coordenadas son una proporción del tamaño de la imagen global. Para obtener más información, consulte. BoundingBox `DetectCustomLabels`devuelve los cuadros delimitadores solo si el modelo está entrenado para detectar la ubicación de los objetos.
- La confianza que Etiquetas personalizadas de Amazon Rekognition tiene en la precisión de la etiqueta y del cuadro delimitador.

Para filtrar las etiquetas en función de la confianza de detección, indique un valor para `MinConfidence` que se correspondan con el nivel de confianza deseado. Por ejemplo, si necesita mucha confianza en la predicción, indique un valor alto para `MinConfidence`. Para obtener todas las etiquetas, independientemente de la confianza, indique un valor 0 para `MinConfidence`.

El rendimiento del modelo se estima, en parte, mediante las métricas de recuperación y precisión calculadas durante el entrenamiento del modelo. Para obtener más información, consulte Métricas para evaluar su modelo.

Para aumentar la precisión del modelo, indique un valor más alto para `MinConfidence`. Para obtener más información, consulte Reducción de los falsos positivos (mayor precisión).

Para aumentar la recuperación del modelo, use un valor inferior para `MinConfidence`. Para obtener más información, consulte Reducción de los falsos negativos (mejor exhaustividad).

Si no indica un valor para `MinConfidence`, Etiquetas personalizadas de Amazon Rekognition devolverá una etiqueta en función del umbral supuesto para esa etiqueta. Para obtener más información, consulte Umbral supuesto. Puede obtener el valor del umbral supuesto para una etiqueta a partir de los resultados de entrenamiento del modelo. Para obtener más información, consulte Entrenamiento de un modelo (consola).

Al usar el parámetro de entrada `MinConfidence`, estará indicando el umbral deseado para la llamada. Las etiquetas detectadas con una confianza inferior al valor de `MinConfidence` no se devuelven en la respuesta. Además, el umbral supuesto de una etiqueta no afecta a la inclusión de la etiqueta en la respuesta.

> **Note**
>
> Las métricas de Etiquetas personalizadas de Amazon Rekognition expresan un umbral supuesto como un valor de punto flotante entre 0 y 1. El rango de `MinConfidence` regula el umbral a un valor porcentual (0-100). Las respuestas de confianza de también DetectCustomLabels se devuelven como un porcentaje.

Puede que desee indicar un umbral para etiquetas concretas. Por ejemplo, si la métrica de precisión es aceptable para la etiqueta A, pero no para la etiqueta B. Al indicar un umbral diferente (`MinConfidence`), tenga en cuenta lo siguiente.

- Si solo le interesa una etiqueta (A), defina el valor de `MinConfidence` en el umbral deseado. En la respuesta, las predicciones de la etiqueta A se devuelven (junto con otras etiquetas) solo si la confianza es superior a `MinConfidence`. Debe filtrar cualquier otra etiqueta que se devuelva.
- Si desea aplicar distintos umbrales a varias etiquetas, haga lo siguiente:
  1. Utilice un valor de 0 para `MinConfidence`. Un valor 0 garantiza que se devuelvan todas las etiquetas, independientemente de la confianza de la detección.
  2. Por cada etiqueta devuelta, aplique el umbral deseado comprobando que la confianza en la etiqueta es superior al umbral que desea para la etiqueta.

Para obtener más información, consulte Mejora de un modelo de Etiquetas personalizadas de Amazon Rekognition.

Si observa que los valores de confianza devueltos por `DetectCustomLabels` son demasiado bajos, considere la posibilidad de volver a entrenar el modelo. Para obtener más información, consulte Entrenamiento de un modelo de Etiquetas personalizadas de Amazon Rekognition. Puede restringir el número de etiquetas personalizadas devueltas a partir de `DetectCustomLabels` indicando el parámetro de entrada `MaxResults`. Los resultados se devuelven ordenados de la confianza más alta a la más baja.

Para ver otros ejemplos que llamen a `DetectCustomLabels`, consulte Ejemplos de etiquetas personalizadas.

Para obtener información sobre la seguridad de `DetectCustomLabels`, consulte Asegurando DetectCustomLabels.

Cómo detectar etiquetas personalizadas (API)

1. Si aún no lo ha hecho:

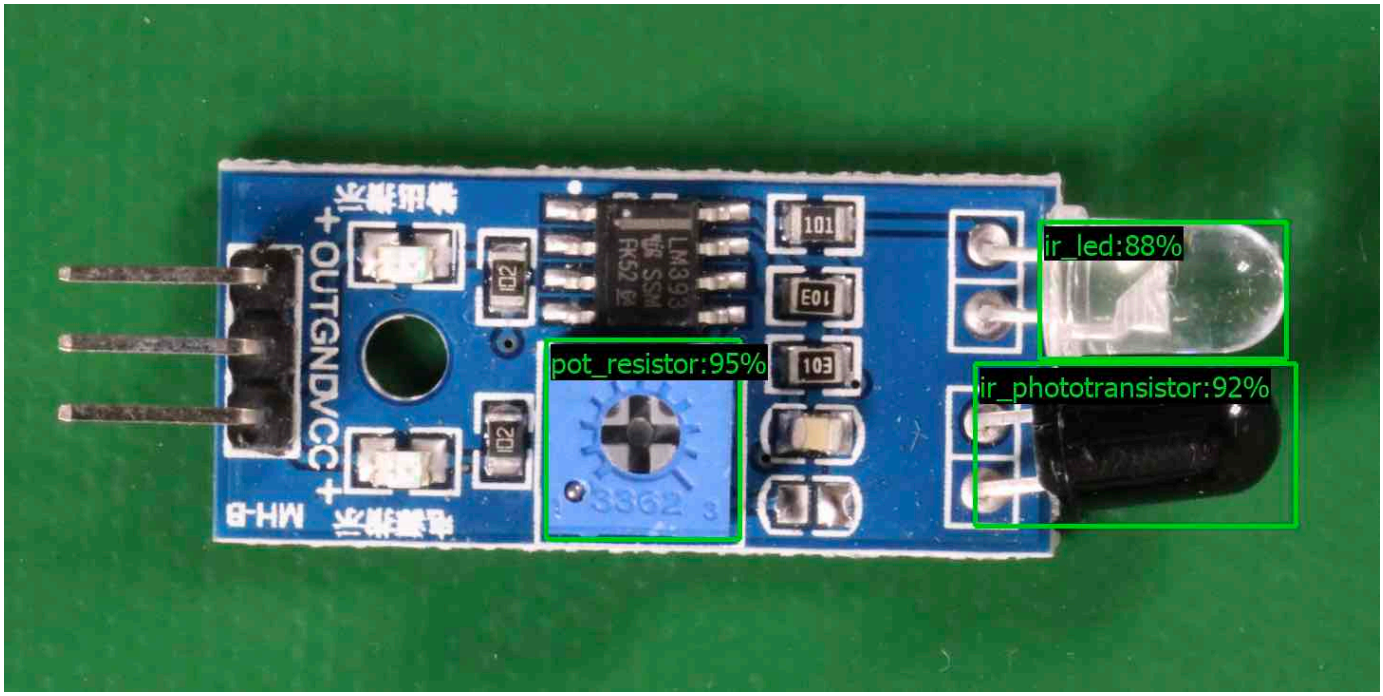
   a. Asegúrese de que tiene los permisos `DetectCustomLabels` y `AmazonS3ReadOnlyAccess`. Para obtener más información, consulte Configuración de permisos de SDK.

   b. Instale y configure el AWS CLI y el AWS SDKs. Para obtener más información, consulte Paso 4: Configure y AWS CLIAWS SDKs.

2. Entrene e implemente su modelo. Para obtener más información, consulte Creación de un modelo de Etiquetas personalizadas de Amazon Rekognition.
3. Asegúrese de que el usuario que llama a `DetectCustomLabels` tenga acceso al modelo utilizado en el paso 2. Para obtener más información, consulte Asegurando DetectCustomLabels.
4. Cargue una imagen que desee analizar en un bucket de S3.

   Para ver las instrucciones, consulte Carga de objetos en Amazon S3 en la Guía del usuario de Amazon Simple Storage Service. Los ejemplos de Python, Java y Java 2 también indican cómo usar un archivo de imagen local para pasar una imagen mediante bytes sin procesar. El archivo debe tener un tamaño inferior a 4 MB.
5. Consulte los siguientes ejemplos para llamar a la operación `DetectCustomLabels`. En los ejemplos de Python y Java se ve la imagen y se superponen los resultados del análisis, de

forma similar a la imagen siguiente. Las siguientes imágenes contienen cuadros delimitadores y etiquetas para una placa de circuito con un potenciómetro, un fototransistor de infrarrojos y componentes LED.



AWS CLI

Este AWS CLI comando muestra el resultado JSON de la operación `DetectCustomLabels` CLI. Cambie los valores de los siguientes parámetros de entrada.

- `bucket` por el nombre del bucket de Amazon S3 utilizado en el paso 4.
- `image` por el nombre del archivo de imagen de entrada que se cargó en el paso 4.
- `projectVersionArn` por el ARN del modelo que desea utilizar.

```
aws rekognition detect-custom-labels --project-version-arn model_arn \
   --image '{"S3Object":{"Bucket":"bucket","Name":"image"}}' \
   --min-confidence 70 \
   --profile custom-labels-access
```

Python

En el siguiente código de ejemplo se muestran los cuadros delimitadores y las etiquetas de nivel de imagen que se encuentran en una imagen.

Para analizar una imagen local, ejecute el programa e indique los siguientes argumentos de línea de comandos:

- El ARN del modelo con el que desea analizar la imagen.
- El nombre y la ubicación del archivo de una imagen local.

Para analizar una imagen almacenada en un bucket de Amazon S3, ejecute el programa e indique los siguientes argumentos de línea de comandos:

- El ARN del modelo con el que desea analizar la imagen.
- El nombre y la ubicación de una imagen con el bucket de Amazon S3 utilizado en el paso 4.
- `--bucket`*`bucket name`*— El bucket de Amazon S3 que utilizó en el paso 4.

Tenga en cuenta que en este ejemplo se supone que su versión de Pillow es >= 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

En el siguiente código de ejemplo se muestran los cuadros delimitadores y las etiquetas de nivel de imagen que se encuentran en una imagen.

Para analizar una imagen local, ejecute el programa e indique los siguientes argumentos de línea de comandos:

- El ARN del modelo con el que desea analizar la imagen.
- El nombre y la ubicación del archivo de una imagen local.

Para analizar una imagen almacenada en un bucket de Amazon S3, ejecute el programa e indique los siguientes argumentos de línea de comandos:

- El ARN del modelo con el que desea analizar la imagen.
- El nombre y la ubicación de una imagen con el bucket de Amazon S3 utilizado en el paso 4.
- El bucket de Amazon S3 que contiene la imagen utilizada en el paso 4.

```
/*
   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

En el siguiente código de ejemplo se muestran los cuadros delimitadores y las etiquetas de nivel de imagen que se encuentran en una imagen.

Para analizar una imagen local, ejecute el programa e indique los siguientes argumentos de línea de comandos:

- `projectVersionArn`: el ARN del modelo con el que desea analizar la imagen.
- `photo`: el nombre y la ubicación del archivo de una imagen local.

Para analizar una imagen almacenada en un bucket de S3, ejecute el programa e indique los siguientes argumentos de línea de comandos:

- El ARN del modelo con el que desea analizar la imagen.
- El nombre y la ubicación de una imagen en el bucket de S3 utilizado en el paso 4.
- El bucket de Amazon S3 que contiene la imagen utilizada en el paso 4.

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

package com.example.rekognition;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.core.ResponseBytes;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.core.sync.ResponseTransformer;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.S3Object;
import software.amazon.awssdk.services.rekognition.model.Image;
import
 software.amazon.awssdk.services.rekognition.model.DetectCustomLabelsRequest;
import
 software.amazon.awssdk.services.rekognition.model.DetectCustomLabelsResponse;
import software.amazon.awssdk.services.rekognition.model.CustomLabel;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import software.amazon.awssdk.services.rekognition.model.BoundingBox;

import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.GetObjectResponse;
import software.amazon.awssdk.services.s3.model.NoSuchBucketException;
import software.amazon.awssdk.services.s3.model.NoSuchKeyException;

import java.io.ByteArrayInputStream;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.io.InputStream;
import java.util.List;

import java.awt.*;
```

```
import java.awt.font.FontRenderContext;
import java.awt.image.BufferedImage;
import javax.imageio.ImageIO;
import javax.swing.*;

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

// Calls DetectCustomLabels on an image. Displays bounding boxes or
// image level labels found in the image.
public class ShowCustomLabels extends JPanel {

    private transient BufferedImage image;
    private transient DetectCustomLabelsResponse response;
    private transient Dimension dimension;
    public static final Logger logger =
 Logger.getLogger(ShowCustomLabels.class.getName());

    // Finds custom labels in an image stored in an S3 bucket.
    public ShowCustomLabels(RekognitionClient rekClient,
            S3Client s3client,
            String projectVersionArn,
            String bucket,
            String key,
            Float minConfidence) throws RekognitionException,
 NoSuchBucketException, NoSuchKeyException, IOException {

        logger.log(Level.INFO, "Processing S3 bucket: {0} image {1}", new
 Object[] { bucket, key });
        // Get image from S3 bucket and create BufferedImage
        GetObjectRequest requestObject =
 GetObjectRequest.builder().bucket(bucket).key(key).build();
        ResponseBytes<GetObjectResponse> result =
 s3client.getObject(requestObject, ResponseTransformer.toBytes());
        ByteArrayInputStream bis = new
 ByteArrayInputStream(result.asByteArray());
        image = ImageIO.read(bis);

        // Set image size
        setWindowDimensions();

        // Construct request parameter for DetectCustomLabels
        S3Object s3Object = S3Object.builder().bucket(bucket).name(key).build();
```

```
        Image s3Image = Image.builder().s3Object(s3Object).build();

        DetectCustomLabelsRequest request =
DetectCustomLabelsRequest.builder().image(s3Image)

.projectVersionArn(projectVersionArn).minConfidence(minConfidence).build();

        response = rekClient.detectCustomLabels(request);
        logFoundLabels(response.customLabels());
        drawLabels();

    }

    // Finds custom label in a local image file.
    public ShowCustomLabels(RekognitionClient rekClient,
            String projectVersionArn,
            String photo,
            Float minConfidence)
            throws IOException, RekognitionException {

        logger.log(Level.INFO, "Processing local file: {0}", photo);
        // Get image bytes and buffered image
        InputStream sourceStream = new FileInputStream(new File(photo));
        SdkBytes imageBytes = SdkBytes.fromInputStream(sourceStream);
        ByteArrayInputStream inputStream = new
ByteArrayInputStream(imageBytes.asByteArray());
        image = ImageIO.read(inputStream);

        setWindowDimensions();

        // Construct request parameter for DetectCustomLabels
        Image localImageBytes = Image.builder().bytes(imageBytes).build();

        DetectCustomLabelsRequest request =
DetectCustomLabelsRequest.builder().image(localImageBytes)

.projectVersionArn(projectVersionArn).minConfidence(minConfidence).build();

        response = rekClient.detectCustomLabels(request);

        logFoundLabels(response.customLabels());
        drawLabels();

    }
```

```
    // Sets window dimensions to 1/2 screen size, unless image is smaller
    public void setWindowDimensions() {
        dimension = java.awt.Toolkit.getDefaultToolkit().getScreenSize();

        dimension.width = (int) dimension.getWidth() / 2;
        if (image.getWidth() < dimension.width) {
            dimension.width = image.getWidth();
        }
        dimension.height = (int) dimension.getHeight() / 2;

        if (image.getHeight() < dimension.height) {
            dimension.height = image.getHeight();
        }

        setPreferredSize(dimension);

    }

    // Draws bounding boxes (if present) and label text.
    public void drawLabels() {

        int boundingBoxBorderWidth = 5;
        int imageHeight = image.getHeight(this);
        int imageWidth = image.getWidth(this);

        // Set up drawing
        Graphics2D g2d = image.createGraphics();
        g2d.setColor(Color.GREEN);
        g2d.setFont(new Font("Tahoma", Font.PLAIN, 50));
        Font font = g2d.getFont();
        FontRenderContext frc = g2d.getFontRenderContext();
        g2d.setStroke(new BasicStroke(boundingBoxBorderWidth));

        List<CustomLabel> customLabels = response.customLabels();

        int imageLevelLabelHeight = 0;
        for (CustomLabel customLabel : customLabels) {

            String label = customLabel.name();

            int textWidth = (int) (font.getStringBounds(label, frc).getWidth());
            int textHeight = (int) (font.getStringBounds(label,
frc).getHeight());
```

```
            // Draw bounding box, if present
            if (customLabel.geometry() != null) {

                BoundingBox box = customLabel.geometry().boundingBox();
                float left = imageWidth * box.left();
                float top = imageHeight * box.top();

                // Draw black rectangle
                g2d.setColor(Color.BLACK);
                g2d.fillRect(Math.round(left + (boundingBoxBorderWidth)),
Math.round(top + (boundingBoxBorderWidth)),
                        textWidth + boundingBoxBorderWidth, textHeight +
boundingBoxBorderWidth);

                // Write label onto black rectangle
                g2d.setColor(Color.GREEN);
                g2d.drawString(label, left + boundingBoxBorderWidth, (top +
textHeight));

                // Draw bounding box around label location
                g2d.drawRect(Math.round(left), Math.round(top),
Math.round((imageWidth * box.width())),
                        Math.round((imageHeight * box.height())));
            }
            // Draw image level labels.
            else {
                // Draw black rectangle
                g2d.setColor(Color.BLACK);
                g2d.fillRect(10, 10 + imageLevelLabelHeight, textWidth,
textHeight);
                g2d.setColor(Color.GREEN);
                g2d.drawString(label, 10, textHeight + imageLevelLabelHeight);

                imageLevelLabelHeight += textHeight;
            }

        }
        g2d.dispose();

    }

    // Log the labels found by DetectCustomLabels
    private void logFoundLabels(List<CustomLabel> customLabels) {
```

```
        logger.info("Custom labels found:");
        if (customLabels.isEmpty()) {
            logger.log(Level.INFO, "No Custom Labels found. Consider lowering
 min confidence.");

        }
        else {
        for (CustomLabel customLabel : customLabels) {
            logger.log(Level.INFO, " Label: {0} Confidence: {1}",
                    new Object[] { customLabel.name(),
 customLabel.confidence() } );
            }
        }
    }

    // Draws the image containing the bounding boxes and labels.
    @Override
    public void paintComponent(Graphics g) {

        Graphics2D g2d = (Graphics2D) g; // Create a Java2D version of g.

        // Draw the image.
        g2d.drawImage(image, 0, 0, dimension.width, dimension.height, this);

    }

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

        String photo = null;
        String bucket = null;
        String projectVersionArn = null;

        final String USAGE = "\n" + "Usage: " + "<model_arn> <image> <bucket>\n
\n" + "Where:\n"
                + "   model_arn - The ARN of the model that you want to use. \n
\n"
                + "   image - The location of the image on your local file
 system or within an S3 bucket.\n\n"
                + "   bucket - The S3 bucket that contains the image. Don't
 specify if image is local.\n\n";

        // Collect the arguments. If 3 arguments are present, the image is
 assumed to be
        // in an S3 bucket.
```

```
        if (args.length < 2 || args.length > 3) {
            System.out.println(USAGE);
            System.exit(1);
        }

        projectVersionArn = args[0];
        photo = args[1];

        if (args.length == 3) {
            bucket = args[2];
        }

        float minConfidence = 50;

        ShowCustomLabels panel = null;

        try {
            // Get the Rekognition client


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

            S3Client s3Client = S3Client.builder()
            .credentialsProvider(ProfileCredentialsProvider.create("custom-
labels-access"))
            .region(Region.US_WEST_2)
            .build();


            // Create frame and panel.
            JFrame frame = new JFrame("Custom Labels");
            frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

            if (args.length == 2) {
                // Analyze local image
                panel = new ShowCustomLabels(rekClient, projectVersionArn,
 photo, minConfidence);
            } else {
```

```
                // Analyze image in S3 bucket
                panel = new ShowCustomLabels(rekClient, s3Client,
 projectVersionArn, bucket, photo, minConfidence);
            }

            frame.setContentPane(panel);
            frame.pack();
            frame.setVisible(true);

        } catch (RekognitionException rekError) {

            String errorMessage = "Rekognition client error: " +
 rekError.getMessage();
            logger.log(Level.SEVERE, errorMessage);
            System.out.println(errorMessage);
            System.exit(1);
        } catch (FileNotFoundException fileError) {
            String errorMessage = "File not found: " + photo;
            logger.log(Level.SEVERE, errorMessage);
            System.out.println(errorMessage);
            System.exit(1);
        } catch (IOException fileError) {
            String errorMessage = "Input output exception: " +
 fileError.getMessage();
            logger.log(Level.SEVERE, errorMessage);
            System.out.println(errorMessage);
            System.exit(1);
        } catch (NoSuchKeyException bucketError) {
            String errorMessage = String.format("Image not found: %s in bucket
 %s.", photo, bucket);
            logger.log(Level.SEVERE, errorMessage);
            System.out.println(errorMessage);
            System.exit(1);
        } catch (NoSuchBucketException bucketError) {
            String errorMessage = "Bucket not found: " + bucket;
            logger.log(Level.SEVERE, errorMessage);
            System.out.println(errorMessage);
            System.exit(1);
        }

    }
}
```

DetectCustomLabels solicitud de operación

En la operación `DetectCustomLabels`, indique una imagen de entrada como matriz de bytes codificada en base64 o como una imagen almacenada en un bucket de Amazon S3. En la siguiente solicitud JSON de ejemplo, aparece la imagen cargada desde un bucket de Amazon S3.

```
{
    "ProjectVersionArn": "string",
     "Image":{
        "S3Object":{
            "Bucket":"string",
            "Name":"string",
            "Version":"string"
         }
    },
    "MinConfidence": 90,
    "MaxLabels": 10,
}
```

DetectCustomLabels respuesta de operación

En la siguiente respuesta JSON de la operación `DetectCustomLabels` se incluyen las palabras y las líneas detectadas en la imagen siguiente.

```
{
    "CustomLabels": [
        {
            "Name": "MyLogo",
            "Confidence": 77.7729721069336,
            "Geometry": {
                "BoundingBox": {
                    "Width": 0.198987677693367,
                    "Height": 0.31296101212501526,
                    "Left": 0.07924537360668182,
                    "Top": 0.4037395715713501
                }
            }
        }
    ]
}
```

Administración de los recursos de Etiquetas personalizadas de Amazon Rekognition

Esta sección ofrece un resumen sobre los recursos de Etiquetas personalizadas de Amazon Rekognition. que puede usar para entrenar y administrar un modelo. También se incluye información general sobre el uso del AWS SDK para entrenar y usar un modelo.

Las Etiquetas personalizadas de Amazon Rekognition se basan en tres recursos diferentes para detectar sus etiquetas personalizadas: proyectos, conjuntos de datos y modelos.

- Proyectos: se utilizan para agrupar otros recursos, como conjuntos de datos, versiones de modelos y evaluaciones de modelos.
- Conjuntos de datos: define las imágenes y los metadatos asociados para su uso en modelos de entrenamiento y prueba. Puede crear un conjunto de datos mediante un archivo de manifiesto con formato SageMaker AI o copiando un conjunto de datos de etiquetas personalizadas de Amazon Rekognition existente.
- Modelos: modelo matemático que predice realmente la presencia de objetos, escenas y conceptos en las imágenes mediante la identificación de patrones en las imágenes utilizadas para entrenar el modelo.

Temas

Administración de un proyecto de Etiquetas personalizadas de Amazon Rekognition

En Etiquetas personalizadas de Amazon Rekognition, un proyecto sirve para gestionar los modelos que cree para una aplicación práctica específica. El proyecto se encarga de administrar los conjuntos de datos, el entrenamiento de los modelos, las versiones de los modelos, la evaluación de los modelos y el funcionamiento de los modelos del proyecto.

Temas

Eliminación de un proyecto de Etiquetas personalizadas de Amazon Rekognition

Puede eliminar un proyecto mediante la consola Amazon Rekognition o llamando a la API. DeleteProject Para eliminar un proyecto, primero debe eliminar todos los modelos asociados. No se puede recuperar un proyecto o modelo después de eliminarlo.

Temas

Eliminación de un proyecto de Etiquetas personalizadas de Amazon Rekognition (consola)

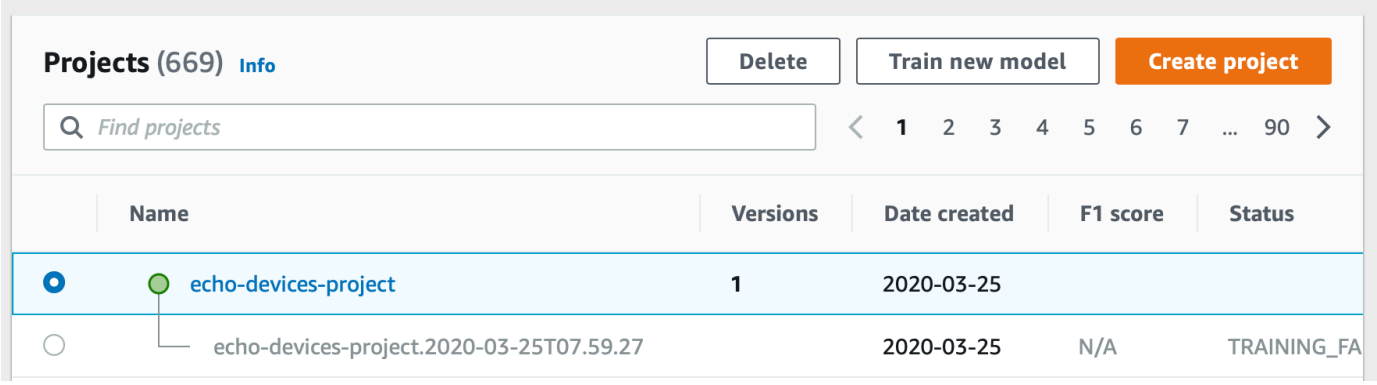
Puede eliminar un proyecto en la página de proyectos o en la página de detalles de un proyecto. En el siguiente procedimiento, se explica cómo eliminar un proyecto en la página de proyectos.

La consola de Etiquetas personalizadas de Amazon Rekognition elimina los modelos y conjuntos de datos asociados mientras se elimina el proyecto. No es posible eliminar un proyecto si alguno de los modelos está en ejecución o en fase de entrenamiento. Para detener un modelo en ejecución, consulte Detención de un modelo de Etiquetas personalizadas de Amazon Rekognition (SDK). Si el modelo se está entrenando, espere a que termine antes de eliminar el proyecto.

Cómo eliminar un proyecto (consola)

1. Abra la consola Amazon Rekognition en. https://console.aws.amazon.com/rekognition/
2. Elija Usar etiquetas personalizadas.
3. Elija Comenzar.
4. En el panel de navegación izquierdo, elija Proyectos.

5. En la página Proyectos, seleccione el botón que aparece junto al proyecto que desea eliminar. Se muestra la lista de proyectos echo-devices-project, con una versión creada el 25 de marzo de 2020 y las opciones para eliminar, entrenar un nuevo modelo o crear un proyecto.



6. En la parte superior de la página, elija Eliminar. Tras esto, se abrirá el cuadro de diálogo Eliminar proyecto.
7. En caso de que el proyecto no tenga modelos asociados:

   a. Escriba eliminar para eliminar el proyecto.

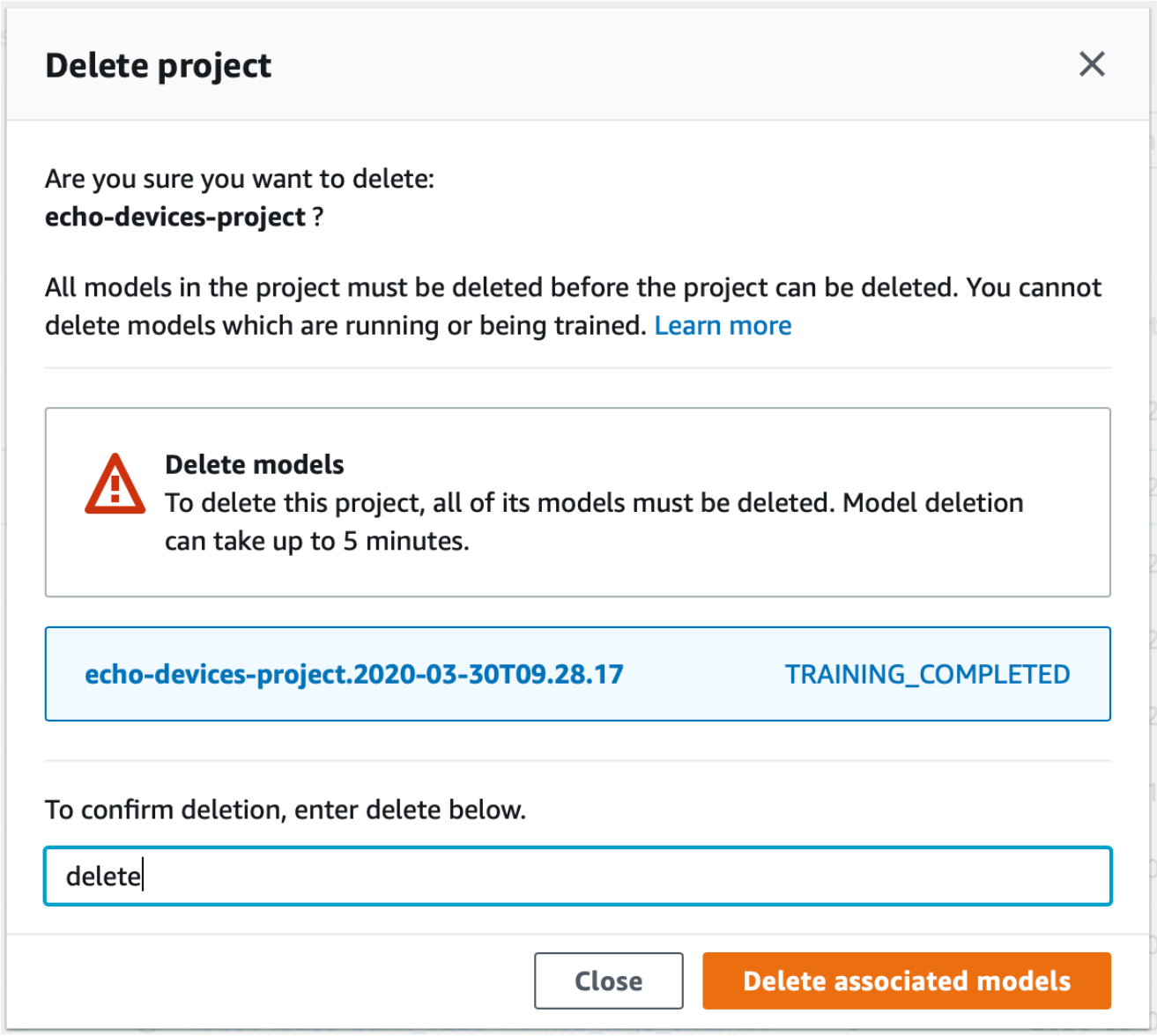
   b. Seleccione Eliminar para eliminar el modelo.
8. En caso de que el proyecto tenga modelos asociados:

   a. Escriba eliminar para confirmar que desea eliminar los modelos y conjuntos de datos.

   b. Elija Eliminar modelos asociados, Eliminar conjuntos de datos asociados o Eliminar conjuntos de datos y modelos asociados, en función de si el modelo tiene conjuntos de datos, modelos o ambos. El proceso de eliminación del modelo puede tardar un tiempo.

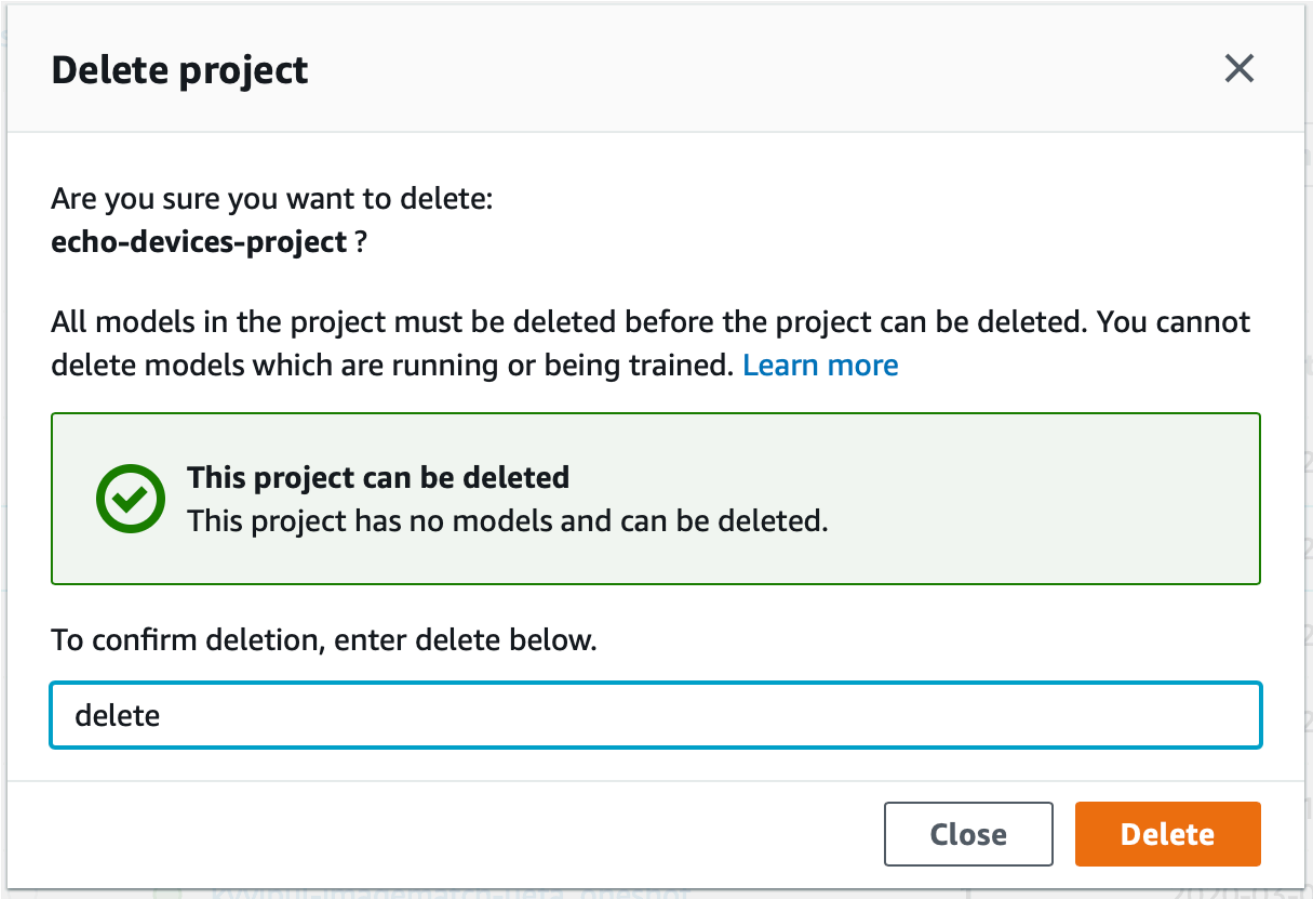
> **Note**
>
> La consola no puede eliminar modelos que estén en fase de entrenamiento o en ejecución. Inténtelo de nuevo después de detener los modelos en ejecución que aparecen en la lista y espere a que los modelos que se estén entrenando terminen. Si cierra el cuadro de diálogo mientras se elimina el modelo, los modelos seguirán borrándose. En otro momento, podrá eliminar el proyecto repitiendo este procedimiento.

   El panel para eliminar un modelo proporciona instrucciones explícitas para eliminar los modelos asociados.



c. Escriba eliminar para confirmar que quiere eliminar el proyecto.

d. Seleccione Eliminar para eliminar el proyecto.



Eliminación de un proyecto de Etiquetas personalizadas de Amazon Rekognition (SDK)

Para eliminar un proyecto de Amazon Rekognition Custom Labels, debe DeleteProjectllamar y proporcionar el nombre de recurso de Amazon (ARN) del proyecto que desea eliminar. Para obtener los proyectos ARNs de su AWS cuenta, llame. DescribeProjects La respuesta incluye un conjunto de ProjectDescriptionobjetos. El ARN del proyecto se corresponde con el campo `ProjectArn`. Puede usar el nombre del proyecto para identificar el ARN del proyecto. Por ejemplo, `arn:aws:rekognition:us-east-1:123456789010:project/`*`project name`*`/1234567890123`.

Antes de eliminar un proyecto, primero debe eliminar todos los modelos y conjuntos de datos del proyecto. Para obtener más información, consulte Eliminación de un modelo de Etiquetas personalizadas de Amazon Rekognition (SDK) y Eliminación de un conjuntos de datos.

Puede que el proyecto tarde unos minutos en eliminarse. Durante ese tiempo, el estado del proyecto será `DELETING`. El proyecto se elimina si en una llamada posterior a DescribeProjects no incluye el proyecto que ha eliminado.

Para eliminar un proyecto (SDK)

1. Si aún no lo ha hecho, instale y configure el AWS CLI y el AWS SDKs. Para obtener más información, consulte Paso 4: Configure y AWS CLIAWS SDKs.
2. Utilice el código siguiente para eliminar un proyecto.

   AWS CLI

   Cambie el valor de `project-arn` por el nombre del proyecto que desee eliminar.

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

   Python

   Utilice el siguiente código. Indique los siguientes parámetros de línea de comandos:

   - `project_arn`: el ARN del proyecto que desea eliminar.

```
# 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

Utilice el siguiente código. Indique los siguientes parámetros de línea de comandos:

- `project_arn`: el ARN del proyecto que desea eliminar.

```
/*
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);
        }
    }

}
```

Descripción de un proyecto (SDK)

Puede utilizar la API de `DescribeProjects` para obtener información sobre sus proyectos.

Cómo describir un proyecto (SDK)

1. Si aún no lo ha hecho, instale y configure el AWS CLI y el AWS SDKs. Para obtener más información, consulte Paso 4: Configure y AWS CLIAWS SDKs.
2. Utilice el siguiente código de ejemplo para describir un proyecto. Cambie `project_name` por el nombre del proyecto que desee describir. Si no indica `--project-names`, se devolverán las descripciones de todos los proyectos.

AWS CLI

```
aws rekognition describe-projects --project-names project_name \
  --profile custom-labels-access
```

Python

Utilice el siguiente código. Indique los siguientes parámetros de línea de comandos:

- project_name: el nombre del proyecto que desee describir. Si no indica un nombre, se devolverán las descripciones de todos los proyectos.

```
# 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

Utilice el siguiente código. Indique los siguientes parámetros de línea de comandos:

- `project_name`: el ARN del proyecto que desea describir. Si no indica un nombre, se devolverán las descripciones de todos los proyectos.

```
/*
   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);
        }

    }
```

```
}
```

Creación de un proyecto con AWS CloudFormation

Amazon Rekognition Custom Labels está AWS CloudFormation integrado con un servicio que le ayuda a modelar y AWS configurar sus recursos para que pueda dedicar menos tiempo a crear y administrar sus recursos e infraestructura. Usted crea una plantilla que describe todos los AWS recursos que desea y AWS CloudFormation se encarga de aprovisionar y configurar esos recursos por usted.

Se puede utilizar AWS CloudFormation para aprovisionar y configurar proyectos de etiquetas personalizadas de Amazon Rekognition.

Cuando la utilices AWS CloudFormation, podrás reutilizar la plantilla para configurar tus proyectos de Amazon Rekognition Custom Labels de forma coherente y repetida. Simplemente describa sus proyectos una vez y, a continuación, aprovisione los mismos proyectos una y otra vez en varias AWS cuentas y regiones.

Etiquetas y plantillas personalizadas de Amazon Rekognition AWS CloudFormation

Para aprovisionar y configurar proyectos de Etiquetas personalizadas de Amazon Rekognition y los servicios relacionados, debe conocer el funcionamiento de las plantillas de AWS CloudFormation. Las plantillas son archivos de texto con formato JSON o YAML. Estas plantillas describen los recursos que desea aprovisionar en sus pilas. AWS CloudFormation Si no estás familiarizado con JSON o YAML, puedes usar AWS CloudFormation Designer para ayudarte a empezar con AWS CloudFormation las plantillas. Para obtener más información, consulte ¿Qué es Designer de AWS CloudFormation ? en la Guía del usuario de AWS CloudFormation .

Para obtener información de referencia sobre los proyectos de Etiquetas personalizadas de Amazon Rekognition, incluidos ejemplos de plantillas JSON y YAML, consulte Referencia de tipo de recurso de Rekognition.

Más información sobre AWS CloudFormation

Para obtener más información AWS CloudFormation, consulte los siguientes recursos:

- AWS CloudFormation
- AWS CloudFormation Guía del usuario

- AWS CloudFormation Referencia de la API
- AWS CloudFormation Guía del usuario de la interfaz de línea de comandos

Administración de conjuntos de datos

Los conjuntos de datos contienen las imágenes y las etiquetas asignadas que se utilizan para entrenar y probar un modelo. Los temas de esta sección muestran cómo administrar un conjunto de datos con la consola Amazon Rekognition Custom Labels y el SDK. AWS

Temas

- Agregar un conjunto de datos a un proyecto
- Agregar más imágenes a un conjunto de datos
- Creación de un conjunto de datos mediante un conjunto de datos existente (SDK)
- Descripción de un conjunto de datos (SDK)
- Listado de entradas del conjunto de datos (SDK)
- Distribución de un conjunto de datos de entrenamiento (SDK)
- Eliminación de un conjuntos de datos

Agregar un conjunto de datos a un proyecto

Puede agregar un conjunto de datos de entrenamiento o un conjunto de datos de prueba a un proyecto existente. Si quiere reemplazar un conjunto de datos existente, borre primero el conjunto de datos existente. Para obtener más información, consulte Eliminación de un conjuntos de datos. A continuación, agregue el nuevo conjunto de datos.

Temas

- Agregar un conjunto de datos a un proyecto (consola)
- Agregar un conjunto de datos a un proyecto (SDK)

Agregar un conjunto de datos a un proyecto (consola)

Puede agregar un conjunto de datos de entrenamiento o de prueba a un proyecto mediante la consola de Etiquetas personalizadas de Amazon Rekognition.

Agregar un conjunto de datos a un proyecto

1. Abra la consola Amazon Rekognition en. https://console.aws.amazon.com/rekognition/
2. En el panel izquierdo, elija Usar etiquetas personalizadas. Se abrirá la página de inicio de Etiquetas personalizadas de Amazon Rekognition.
3. En el panel de navegación izquierdo, elija Proyectos. Se abrirá la vista de proyectos.
4. Elija el proyecto al que desee agregar un conjunto de datos.
5. En el panel de navegación izquierdo, en el nombre del proyecto, elija Conjuntos de datos.
6. Si el proyecto no tiene un conjunto de datos existente, se abrirá la página Crear conjunto de datos. Haga lo siguiente:

   a. En la página Crear conjunto de datos, introduzca la información de la fuente de la imagen. Para obtener más información, consulte the section called “Crear conjuntos de datos con imágenes”.

   b. Elija Crear conjunto de datos para crear el conjunto de datos.
7. Si el proyecto tiene un conjunto de datos existente (de entrenamiento o de prueba), se abrirá la página de detalles del proyecto. Haga lo siguiente:

   a. En la página de detalles del proyecto, seleccione Acciones.

   b. Si quiere agregar un conjunto de datos de entrenamiento, elija Crear conjunto de datos de entrenamiento.

   c. Si quiere agregar un conjunto de datos de prueba, elija Crear conjunto de datos de prueba.

   d. En la página Crear conjunto de datos, introduzca la información de la fuente de la imagen. Para obtener más información, consulte the section called “Crear conjuntos de datos con imágenes”.

   e. Elija Crear conjunto de datos para crear el conjunto de datos.
8. Añada imágenes al conjunto de datos. Para obtener más información, consulte Agregar más imágenes (consola).
9. Añada etiquetas al conjunto de datos. Para obtener más información, consulte Agregar etiquetas nuevas (consola).
10. Añada etiquetas a las imágenes. Si va a agregar etiquetas de imagen, consulte the section called “Asignación de etiquetas de imagen a una imagen”. Si va a agregar cuadros delimitadores, consulte Etiquetado de objetos con cuadros delimitadores. Para obtener más información, consulte Finalidad de los conjuntos de datos.

Agregar un conjunto de datos a un proyecto (SDK)

Puedes agregar un conjunto de datos de entrenamiento o de prueba a un proyecto existente de las siguientes maneras:

- Cree un conjunto de datos a través de un archivo de manifiesto. Para obtener más información, consulte Creación de un conjunto de datos con un archivo de manifiesto (SDK) de SageMaker AI Ground Truth.
- Cree un conjunto de datos vacío y rellénelo después. En el ejemplo siguiente se muestra cómo crear un conjunto de datos vacío. Para agregar entradas después de crear un conjunto de datos vacío, consulte Agregar más imágenes a un conjunto de datos.

Cómo agregar un conjunto de datos a un proyecto (SDK)

1. Si aún no lo ha hecho, instale y configure el y el AWS CLI . AWS SDKs Para obtener más información, consulte Paso 4: Configure y AWS CLIAWS SDKs.
2. Use los siguientes ejemplos para agregar líneas JSON a un conjunto de datos.

   CLI

   Reemplace `project_arn` por el proyecto al que desee agregar el conjunto de datos. Sustituya `dataset_type` por `TRAIN` para crear un conjunto de datos de entrenamiento o por `TEST` para crear un conjunto de datos de prueba.

   ```
   aws rekognition create-dataset --project-arn project_arn \
     --dataset-type dataset_type \
     --profile custom-labels-access
   ```

   Python

   Cree un conjunto de datos con el siguiente código. Indique las siguientes opciones de línea de comandos:

   - `project_arn`: el ARN del proyecto al que desea agregar el conjunto de datos de prueba.
   - `type`: el tipo de conjunto de datos que desea crear (entrenamiento o prueba)

   ```
   # 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

Cree un conjunto de datos con el siguiente código. Indique las siguientes opciones de línea de comandos:

- `project_arn`: el ARN del proyecto al que desea agregar el conjunto de datos de prueba.

- `type`: el tipo de conjunto de datos que desea crear (entrenamiento o prueba)

```
/*
   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. Añada imágenes al conjunto de datos. Para obtener más información, consulte Agregar más imágenes (SDK).

Agregar más imágenes a un conjunto de datos

Puede agregar más imágenes a sus conjuntos de datos mediante la consola de Etiquetas personalizadas de Amazon Rekognition o llamando a la API de `UpdateDatasetEntries`.

Temas

- Agregar más imágenes (consola)
- Agregar más imágenes (SDK)

Agregar más imágenes (consola)

Al usar la consola de Etiquetas personalizadas de Amazon Rekognition, las imágenes se cargan del ordenador local. Las imágenes se agregan a la ubicación del bucket de Amazon S3 (consola o externo) donde se almacenan las imágenes utilizadas para crear el conjunto de datos.

Cómo agregar más imágenes a su conjunto de datos (consola)

1. Abra la consola Amazon Rekognition en. https://console.aws.amazon.com/rekognition/
2. En el panel izquierdo, elija Usar etiquetas personalizadas. Se abrirá la página de inicio de Etiquetas personalizadas de Amazon Rekognition.
3. En el panel de navegación izquierdo, elija Proyectos. Se abrirá la vista de proyectos.
4. Elija el proyecto que desee usar.
5. En el panel de navegación izquierdo, en el nombre del proyecto, elija Conjunto de datos.
6. Elija Acciones y seleccione el conjunto de datos al que desee agregar las imágenes.
7. Escoja las imágenes que quiera cargar en el conjunto de datos. Puede arrastrar las imágenes o elegir las imágenes deseadas de su equipo local. Puede cargar hasta 30 imágenes a la vez.
8. Seleccione Cargar imágenes.
9. Elija Guardar cambios.

10. Etiquete las imágenes. Para obtener más información, consulte Etiquetado de imágenes.

Agregar más imágenes (SDK)

`UpdateDatasetEntries` actualiza o agrega líneas JSON a un archivo de manifiesto. Las líneas JSON se pasan como un objeto de datos codificado en byte64 en el campo `GroundTruth`. Si utiliza un AWS SDK para realizar llamadas`UpdateDatasetEntries`, el SDK codifica los datos automáticamente. Cada línea JSON contiene la información de una sola imagen, como las etiquetas asignadas o la información de los cuadros delimitadores. Por ejemplo:

```
{"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"}}
```

Para obtener más información, consulte Creación de un archivo de manifiesto.

Utilice el campo `source-ref` como clave para identificar las imágenes que desee actualizar. Si el conjunto de datos no contiene un valor de campo `source-ref` igual, la línea JSON se agregará como una imagen nueva.

Cómo agregar más imágenes a un conjunto de datos (SDK)

1. Si aún no lo ha hecho, instale y configure el AWS CLI y el AWS SDKs. Para obtener más información, consulte Paso 4: Configure y AWS CLIAWS SDKs.
2. Use los siguientes ejemplos para agregar líneas JSON a un conjunto de datos.

   CLI

   Reemplace el valor de `GroundTruth` por las líneas JSON que desee utilizar. Debe evitar cualquier carácter especial en la línea JSON.

   ```
   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

Utilice el siguiente código. Indique los siguientes parámetros de línea de comandos:

- dataset_arn: el ARN del conjunto de datos que desea actualizar.
- updates_file: el archivo que contiene las actualizaciones de la línea JSON.

```
# 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: el ARN del conjunto de datos que desea actualizar.
- updates_file: el archivo que contiene las actualizaciones de la línea JSON.

```
/*
   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);
        }

    }

}
```

Creación de un conjunto de datos mediante un conjunto de datos existente (SDK)

El siguiente procedimiento muestra cómo crear un conjunto de datos a partir de un conjunto de datos existente mediante la CreateDatasetoperación.

1. Si aún no lo ha hecho, instale y configure el AWS CLI y el AWS SDKs. Para obtener más información, consulte Paso 4: Configure y AWS CLIAWS SDKs.
2. Use el siguiente código de ejemplo para crear un conjunto de datos copiando otro conjunto de datos.

   AWS CLI

   Cree el conjunto de datos con el siguiente código. Sustituya lo siguiente:

   - `project_arn`: el ARN del proyecto al que desea agregar el conjunto de datos.
   - `dataset_type`: por el tipo de conjunto de datos que desee crear en el proyecto (`TRAIN` o `TEST`).
   - `dataset_arn`: por el ARN del conjunto de datos que desee copiar.

   ```
   aws rekognition create-dataset --project-arn project_arn \
     --dataset-type dataset_type \
     --dataset-source '{ "DatasetArn" : "dataset_arn" }' \
     --profile custom-labels-access
   ```

   Python

   En el siguiente ejemplo, se crea un conjunto de datos a partir de un conjunto de datos existente y se indica el ARN.

   Para ejecutar el programa, indique los siguientes argumentos de línea de comandos:

   - `project_arn`: el ARN del proyecto que desea utilizar.
   - `dataset_type`: el tipo de conjunto de datos del proyecto que desee crear (`train` o `test`).
   - `dataset_arn`: el ARN del conjunto de datos a partir del cual desee crear el conjunto de datos.

   ```
   # 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

En el siguiente ejemplo, se crea un conjunto de datos a partir de un conjunto de datos existente y se indica el ARN.

Para ejecutar el programa, indique los siguientes argumentos de línea de comandos:

- `project_arn`: el ARN del proyecto que desea utilizar.
- `dataset_type`: el tipo de conjunto de datos del proyecto que desee crear (`train` o `test`).
- `dataset_arn`: el ARN del conjunto de datos a partir del cual desee crear el conjunto de datos.

```
/*
   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);
        }

    }

}
```

Descripción de un conjunto de datos (SDK)

Puede utilizar la API de `DescribeDataset` para obtener información sobre un conjunto de datos.

Cómo describir un conjunto de datos (SDK)

1. Si aún no lo ha hecho, instale y configure el AWS CLI y el AWS SDKs. Para obtener más información, consulte Paso 4: Configure y AWS CLIAWS SDKs.
2. Incluya la descripción de un conjunto de datos con el siguiente código.

   AWS CLI

   Cambie el valor de `dataset-arn` por el ARN del conjunto de datos que desee describir.

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

   Python

   Utilice el siguiente código. Indique los siguientes parámetros de línea de comandos:

   - dataset_arn: el ARN del conjunto de datos que desee describir.

```
# 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: el ARN del conjunto de datos que desee describir.

```
/*
   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);
        }

    }

}
```

Listado de entradas del conjunto de datos (SDK)

Puede usar la API de `ListDatasetEntries` para ver las líneas JSON de cada imagen en un conjunto de datos. Para obtener más información, consulte Creación de un archivo de manifiesto.

Cómo ver las entradas del conjunto de datos (SDK)

1. Si aún no lo ha hecho, instale y configure el AWS CLI y el AWS SDKs. Para obtener más información, consulte Paso 4: Configure y AWS CLIAWS SDKs.
2. Usa el siguiente ejemplo para ver las entradas en un conjunto de datos.

   AWS CLI

   Cambie el valor de `dataset-arn` por el ARN del conjunto de datos que desee ver.

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

   Para ver solo las líneas JSON con errores, indique `has-errors`.

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

   Python

   Utilice el siguiente código. Indique los siguientes parámetros de línea de comandos:

   - dataset_arn: el ARN del conjunto de datos que desea ver.
   - show_errors_only: indique `true` si quiere ver solo los errores. Si no, elija `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

Utilice el siguiente código. Indique los siguientes parámetros de línea de comandos:

- dataset_arn: el ARN del conjunto de datos que desea ver.
- show_errors_only: indique `true` si quiere ver solo los errores. Si no, elija `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);
        }

    }

}
```

Distribución de un conjunto de datos de entrenamiento (SDK)

Etiquetas personalizadas de Amazon Rekognition necesita un conjunto de datos de entrenamiento y un conjunto de datos de prueba para entrenar el modelo.

Si utilizas la API, puedes utilizarla para distribuir el DistributeDatasetEntries20% del conjunto de datos de entrenamiento en un conjunto de datos de prueba vacío. Distribuir el conjunto de datos de entrenamiento puede resultar útil si solo tiene un archivo de manifiesto disponible. Use el archivo de manifiesto único para crear su conjunto de datos de entrenamiento. A continuación, cree un conjunto de datos de prueba vacío y use `DistributeDatasetEntries` para rellenar el conjunto de datos de prueba.

> **Note**
>
> Si va a usar la consola de Etiquetas personalizadas de Amazon Rekognition y comienza con un único proyecto de conjunto de datos, Etiquetas personalizadas de Amazon Rekognition divide (distribuye) el conjunto de datos de entrenamiento durante el entrenamiento para crear un conjunto de datos de prueba. El 20 % de las entradas del conjunto de datos de entrenamiento pasan al conjunto de datos de prueba.

Cómo distribuir un conjunto de datos de entrenamiento (SDK)

1. Si aún no lo has hecho, instala y configura el AWS CLI y el AWS SDKs. Para obtener más información, consulte Paso 4: Configure y AWS CLIAWS SDKs.
2. Cree un proyecto. Para obtener más información, consulte Creación de un proyecto de Etiquetas personalizadas de Amazon Rekognition (SDK).
3. Cree su conjunto de datos de entrenamiento. Para obtener más información sobre los conjuntos de datos, consulte Creación de conjuntos de datos de entrenamiento y prueba.
4. Cree un conjunto de datos de prueba vacío.
5. Usa el siguiente código de ejemplo para distribuir el 20 % de las entradas del conjunto de datos de entrenamiento en el conjunto de datos de prueba. Puede obtener los nombres de recursos de Amazon (ARN) de los conjuntos de datos de un proyecto llamando. DescribeProjects Para ver el código de ejemplo, consulte Descripción de un proyecto (SDK).

   AWS CLI

   Cambie el valor de `training_dataset-arn` y `test_dataset_arn` por el ARN de los conjuntos de datos que desee usar.

   ```
   aws rekognition distribute-dataset-entries --datasets ['{"Arn":
    "training_dataset_arn"}, {"Arn": "test_dataset_arn"}'] \
     --profile custom-labels-access
   ```

   Python

   Utilice el siguiente código. Indique los siguientes parámetros de línea de comandos:

   - training_dataset_arn: el ARN del conjunto de datos de entrenamiento del que distribuye las entradas.
   - test_dataset_arn: el ARN del conjunto de datos de prueba en el que distribuye las entradas.

   ```
   # 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

Utilice el siguiente código. Indique los siguientes parámetros de línea de comandos:

- training_dataset_arn: el ARN del conjunto de datos de entrenamiento del que distribuye las entradas.
- test_dataset_arn: el ARN del conjunto de datos de prueba en el que distribuye las entradas.

```
/*
   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);
        }

    }

}
```

Eliminación de un conjuntos de datos

Puede eliminar los conjuntos de datos de entrenamiento y de prueba de un proyecto.

Temas

- Eliminación de un conjunto de datos (consola)
- Eliminación de un conjunto de datos de Etiquetas personalizadas de Amazon Rekognition (SDK)

Eliminación de un conjunto de datos (consola)

Consulte el siguiente procedimiento para eliminar un conjunto de datos. Si queda un conjunto de datos en el proyecto (de entrenamiento o de prueba), se abrirá la página de detalles del proyecto. Si en el proyecto no quedan conjuntos de datos, se abrirá la página Crear conjunto de datos.

Si elimina el conjunto de datos de entrenamiento, deberá crear un nuevo conjunto de datos de entrenamiento para el proyecto antes de poder entrenar un modelo. Para obtener más información, consulte Creación de conjuntos de datos de entrenamiento y prueba.

Si elimina el conjunto de datos de prueba, podrá entrenar un modelo sin crear un nuevo conjunto de datos de prueba. Durante el entrenamiento, el conjunto de datos de entrenamiento se divide para crear un nuevo conjunto de datos de prueba para el proyecto. Al dividir el conjunto de datos de entrenamiento, se reduce la cantidad de imágenes disponibles para el entrenamiento. Para mantener la calidad, recomendamos crear un nuevo conjunto de datos de prueba antes de entrenar un modelo. Para obtener más información, consulte Agregar un conjunto de datos a un proyecto.

Cómo eliminar un conjunto de datos

1. Abra la consola Amazon Rekognition en. https://console.aws.amazon.com/rekognition/

2. En el panel izquierdo, elija Usar etiquetas personalizadas. Se abrirá la página de inicio de Etiquetas personalizadas de Amazon Rekognition.
3. En el panel de navegación izquierdo, elija Proyectos. Se abrirá la vista de proyectos.
4. Elija el proyecto que contenga el conjunto de datos que desea eliminar.
5. En el panel de navegación izquierdo, en el nombre del proyecto, elija Conjunto de datos.
6. Elija Acciones.
7. Para eliminar el conjunto de datos de entrenamiento, seleccione Eliminar conjunto de datos de entrenamiento.
8. Para eliminar el conjunto de datos de prueba, elija Eliminar conjunto de datos de prueba.
9. En el cuadro de diálogo Eliminar conjunto de datos de entrenamiento o prueba, escriba eliminar para confirmar que desea eliminar el conjunto de datos.
10. Elija Eliminar conjunto de datos de entrenamiento o prueba, para eliminarlo.

Eliminación de un conjunto de datos de Etiquetas personalizadas de Amazon Rekognition (SDK)

Para eliminar un conjunto de datos de Amazon Rekognition Custom Labels, debe DeleteDatasetllamar y proporcionar el nombre de recurso de Amazon (ARN) del conjunto de datos que desea eliminar. Para obtener los conjuntos ARNs de datos de entrenamiento y prueba de un proyecto, llame. DescribeProjects La respuesta incluye una serie de ProjectDescriptionobjetos. El conjunto de datos ARNs (`DatasetArn`) y los tipos de conjuntos de datos (`DatasetType`) están en la `Datasets` lista.

Si elimina el conjunto de datos de entrenamiento, tendrá que crear un nuevo conjunto de datos de entrenamiento para el proyecto antes de poder entrenar un modelo. Si elimina el conjunto de datos de prueba, tendrá que crear un nuevo conjunto de datos de prueba antes de poder entrenar un modelo. Para obtener más información, consulte Agregar un conjunto de datos a un proyecto (SDK).

Cómo eliminar un conjunto de datos (SDK)

1. Si aún no lo ha hecho, instale y configure el AWS CLI y el AWS SDKs. Para obtener más información, consulte Paso 4: Configure y AWS CLIAWS SDKs.
2. Elimine un conjunto de datos con el siguiente código.

AWS CLI

Cambie el valor de `dataset-arn` por el ARN del conjunto de datos que desee eliminar.

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

Python

Utilice el siguiente código. Indique los siguientes parámetros de línea de comandos:

- dataset_arn: el ARN del conjunto de datos que desea eliminar.

```
# 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

Utilice el siguiente código. Indique los siguientes parámetros de línea de comandos:

- dataset_arn: el ARN del conjunto de datos que desea eliminar.

```
/*
   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);
        }

    }

}
```

Administración de un modelo de Etiquetas personalizadas de Amazon Rekognition

Un modelo de Etiquetas personalizadas de Amazon Rekognition es un modelo matemático que predice la presencia de objetos, escenas y conceptos en imágenes nuevas. Para ello, busca patrones en las imágenes utilizadas para entrenar el modelo. En esta sección se explica cómo entrenar un modelo, evaluar su rendimiento y realizar mejoras. También le enseñará a hacer que un modelo esté disponible para usarlo y a eliminarlo cuando ya no lo necesite.

Temas

Eliminación de un modelo de Etiquetas personalizadas de Amazon Rekognition

Puede eliminar un modelo mediante la consola Amazon Rekognition Custom Labels o mediante la API. DeleteProjectVersion No puede eliminar un modelo si está en ejecución o si se está entrenando. Para detener un modelo en ejecución, utilice la API. StopProjectVersion Para obtener más información, consulte Detención de un modelo de Etiquetas personalizadas de Amazon Rekognition (SDK). Si el modelo se está entrenando, espere a que termine antes de eliminarlo.

No se puede recuperar un modelo después de eliminarlo.

Temas

Eliminación de un modelo de Etiquetas personalizadas de Amazon Rekognition (consola)

En el siguiente procedimiento, se explica cómo eliminar un modelo de la página de detalles del proyecto. También puede eliminar un modelo en la página de detalles de un modelo.

Cómo eliminar un modelo (consola)

1. Abra la consola Amazon Rekognition en. https://console.aws.amazon.com/rekognition/
2. Elija Usar etiquetas personalizadas.
3. Elija Comenzar.
4. En el panel de navegación izquierdo, elija Proyectos.
5. Elija el proyecto que contenga el modelo que desee eliminar. Se abrirá la página de detalles del proyecto.

6. En la sección Modelos, seleccione los modelos que desee eliminar.

> **Note**
>
> Si no se puede seleccionar el modelo, significa que está ejecutándose o entrenándose y no se puede eliminar. Consulte el campo Estado e inténtelo de nuevo después de detener el modelo en ejecución o espere a que termine el entrenamiento.

7. Seleccione Eliminar modelo y aparecerá el cuadro de diálogo Eliminar modelo.
8. Escriba eliminar para confirmar la eliminación.
9. Seleccione Eliminar para eliminar el modelo. El proceso de eliminación del modelo puede tardar unos momentos en completarse.

> **Note**
>
> Si cierra el cuadro de diálogo mientras se elimina el modelo, los modelos seguirán borrándose.

Eliminación de un modelo de Etiquetas personalizadas de Amazon Rekognition (SDK)

Para eliminar un modelo de Amazon Rekognition Custom Labels, debe DeleteProjectVersionllamar y proporcionar el nombre de recurso de Amazon (ARN) del modelo que desea eliminar. Puede obtener el ARN del modelo en la sección Usar su modelo que hay en la página de detalles del modelo en la consola de Etiquetas personalizadas de Amazon Rekognition. Como alternativa, puede llamar DescribeProjectVersionsy proporcionar lo siguiente.

- El ARN del proyecto (`ProjectArn`) al que está asociado el modelo.
- El nombre de la versión del modelo (`VersionNames`) del modelo.

El ARN del modelo es el `ProjectVersionArn` campo del ProjectVersionDescriptionobjeto, a partir de la `DescribeProjectVersions` respuesta.

No puede eliminar un modelo si se está ejecutando o si se está entrenando. Para determinar si el modelo se está ejecutando o entrenando, llame DescribeProjectVersionsy compruebe el `Status` campo del ProjectVersionDescriptionobjeto del modelo. Para detener un modelo en ejecución, utilice la StopProjectVersionAPI. Para obtener más información, consulte Detención de un modelo

de Etiquetas personalizadas de Amazon Rekognition (SDK). Tiene que esperar a que un modelo termine de entrenarse para poder eliminarlo.

Cómo eliminar un modelo (SDK)

1. Si aún no lo ha hecho, instale y configure el AWS CLI y el AWS SDKs. Para obtener más información, consulte Paso 4: Configure y AWS CLIAWS SDKs.
2. Utilice el siguiente código para eliminar un modelo.

   AWS CLI

   Cambie el valor de `project-version-arn` por el nombre del proyecto que desee eliminar.

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

   Python

   Indique los siguientes parámetros de línea de comandos:

   - `project_arn`: el ARN del proyecto que incluya el modelo que desee eliminar.
   - `model_arn`: el ARN de la versión del modelo que desee eliminar.

   ```
   # 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`: el ARN del proyecto que incluya el modelo que desee eliminar.
- `model_arn`: el ARN de la versión del modelo que desee eliminar.

```
//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);
        }

    }

}
```

Etiquetado de un modelo

Puede usar etiquetas para identificar, organizar, buscar y filtrar sus modelos de Etiquetas personalizadas de Amazon Rekognition. Cada etiqueta es una marca que consta de una clave y un valor definidos por el usuario. Por ejemplo, para determinar cómo se facturan los modelos, etiquételos con una clave de `Cost center` y agregue el número de centro de costes correspondiente como valor. Para obtener más información, consulte Etiquetado de recursos de AWS.

Utilice las etiquetas para:

- Llevar un control de la facturación de un modelo mediante etiquetas de asignación de costes. Para obtener más información, consulte Uso de etiquetas de asignación de costes.
- Acceso de control de un modelo mediante Identity and Access Management (IAM) Para obtener más información, consulte Control de acceso a recursos de AWS mediante etiquetas de recursos.
- Automatice la gestión de modelos. Por ejemplo, puede ejecutar scripts automatizados de inicio o detención que desactivan modelos de desarrollo durante las horas no laborables para reducir costes. Para obtener más información, consulte Ejecución de un modelo de Etiquetas personalizadas de Amazon Rekognition.

Puede etiquetar modelos mediante la consola Amazon Rekognition o mediante la. AWS SDKs

Temas

- Etiquetado de modelos (consola)
- Visualización de etiquetas de modelos
- Etiquetado de modelos (SDK)

Etiquetado de modelos (consola)

Puede usar la consola de Rekognition para agregar etiquetas a los modelos, ver las etiquetas asociadas a un modelo y eliminar etiquetas.

Cómo agregar y eliminar etiquetas

Este procedimiento explica cómo agregar o eliminar etiquetas en un modelo existente. También puede añadir etiquetas a un nuevo modelo cuando esté entrenado. Para obtener más información, consulte Entrenamiento de un modelo de Etiquetas personalizadas de Amazon Rekognition.

Cómo agregar o eliminar etiquetas en un modelo existente mediante la consola

1. Abra la consola Amazon Rekognition en. https://console.aws.amazon.com/rekognition/
2. Elija Comenzar.
3. En el panel de navegación, elija Proyectos.
4. En la página Proyectos, elija el proyecto que contiene el modelo que quiera etiquetar.
5. En el panel de navegación, en el proyecto seleccionado anteriormente, elija Modelos.
6. En la sección Modelos, elija el modelo al que desee añadir una etiqueta.
7. En la página de detalles del modelo, elija la pestaña Etiquetas.
8. En la sección Etiquetas, elija Administrar etiquetas.
9. En la página Agregar etiquetas, elija Siguiente.
10. Introduzca una clave y un valor.
    a. En Clave, escriba el nombre de la clave.
    b. En Valor, introduzca un valor.
11. Para añadir más etiquetas, repita los pasos 9 y 10.
12. (Opcional) Para eliminar una etiqueta, elija Eliminar junto a la etiqueta correspondiente. Si va a eliminar una etiqueta guardada anteriormente, se eliminará al guardar los cambios.
13. Elija Guardar cambios para guardar los cambios.

Visualización de etiquetas de modelos

Puede utilizar la consola de Amazon Rekognition para ver las etiquetas asociadas a un modelo.

Para ver las etiquetas asociadas a todos los modelos de un proyecto, debe usar el AWS SDK. Para obtener más información, consulte Listado de etiquetas de modelos.

Para ver las etiquetas asociadas a un modelo

1. Abra la consola Amazon Rekognition en. https://console.aws.amazon.com/rekognition/
2. Elija Comenzar.
3. En el panel de navegación, elija Proyectos.
4. En la página Proyectos, elija el proyecto que contiene el modelo cuya etiqueta quiera ver.
5. En el panel de navegación, en el proyecto seleccionado anteriormente, elija Modelos.
6. En la sección Modelos, elija el modelo cuya etiqueta quiera ver.
7. En la página de detalles del modelo, elija la pestaña Etiquetas. Las etiquetas aparecerán en la sección Etiquetas.

Etiquetado de modelos (SDK)

Puede usar el SDK para: AWS

- Añadir etiquetas a un nuevo modelo
- Añadir etiquetas a un modelo existente
- Ver las etiquetas asociadas a un modelo
- Eliminar etiquetas de un modelo

Las etiquetas de los siguientes AWS CLI ejemplos tienen el siguiente formato.

```
--tags '{"key1":"value1","key2":"value2"}'
```

También puede utilizar este formato.

```
--tags key1=value1,key2=value2
```

Si no ha instalado el AWS CLI, consultePaso 4: Configure y AWS CLIAWS SDKs.

Cómo agregar etiquetas a un nuevo modelo

Puede añadir etiquetas a un modelo al crearlo mediante la CreateProjectVersionoperación. Indique una o varias etiquetas en el parámetro de entrada `Tags` de la matriz.

```
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
```

Para obtener información sobre cómo crear y entrenar un modelo, consulte Entrenamiento de un modelo (SDK).

Cómo agregar etiquetas a un modelo existente

Para añadir una o más etiquetas a un modelo existente, utilice la TagResourceoperación. Indique el nombre de recurso de Amazon (ARN) del modelo (`ResourceArn`) y las etiquetas (`Tags`) que desea agregar. En el siguiente ejemplo se ve cómo agregar dos etiquetas.

```
aws rekognition tag-resource --resource-arn resource-arn \
  --tags '{"key1":"value1","key2":"value2"}' \
  --profile custom-labels-access
```

Puede obtener el ARN de un modelo llamando. CreateProjectVersion

Listado de etiquetas de modelos

Para enumerar las etiquetas adjuntas a un modelo, utilice la ListTagsForResourceoperación y especifique el ARN del modelo ()`ResourceArn`. El resultado será la asignación de las claves y los valores de las etiquetas que se asocian al modelo concreto.

```
aws rekognition list-tags-for-resource --resource-arn resource-arn \
  --profile custom-labels-access
```

Este resultado se verá en forma de lista de las etiquetas asociadas al modelo.

```
{
    "Tags": {
        "Dept": "Engineering",
        "Name": "Ana Silva Carolina",
        "Role": "Developer"
    }
}
```

Para ver qué modelos de un proyecto tienen una etiqueta específica, llame a `DescribeProjectVersions` para obtener una lista de modelos. A continuación,

llame a `ListTagsForResource` para cada modelo en la respuesta a través de `DescribeProjectVersions`. Revise la respuesta de `ListTagsForResource` para ver si está la etiqueta necesaria.

En el siguiente ejemplo de Python 3, se indica cómo buscar en todos sus proyectos una clave y un valor de etiqueta específicos. El resultado incluye el ARN del proyecto y el ARN del modelo, donde hay una clave coincidente.

Cómo buscar un valor de etiqueta

1. Guarde el siguiente código en un archivo denominado `find_tag.py`.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to find a tag value that's associated with models within
your Amazon Rekognition Custom Labels projects.
"""
import logging
import argparse
import boto3

from botocore.exceptions import ClientError


logger = logging.getLogger(__name__)


def find_tag_in_projects(rekognition_client, key, value):
    """
    Finds Amazon Rekognition Custom Label models tagged with the supplied key and
 key value.
    :param rekognition_client: An Amazon Rekognition boto3 client.
    :param key: The tag key to find.
    :param value: The value of the tag that you want to find.
    return: A list of matching model versions (and model projects) that were found.
    """
    try:

        found_tags = []
        found = False
```

```
        projects = rekognition_client.describe_projects()
        # Iterate through each project and models within a project.
        for project in projects["ProjectDescriptions"]:
            logger.info("Searching project: %s ...", project["ProjectArn"])

            models = rekognition_client.describe_project_versions(
                ProjectArn=(project["ProjectArn"])
            )

            for model in models["ProjectVersionDescriptions"]:
                logger.info("Searching model %s", model["ProjectVersionArn"])

                tags = rekognition_client.list_tags_for_resource(
                    ResourceArn=model["ProjectVersionArn"]
                )

                logger.info(
                    "\tSearching model: %s for tag: %s value: %s.",
                    model["ProjectVersionArn"],
                    key,
                    value,
                )
                # Check if tag exists.

                if key in tags["Tags"]:
                    if tags["Tags"][key] == value:
                        found = True
                        logger.info(
                            "\t\tMATCH: Project: %s: model version %s",
                            project["ProjectArn"],
                            model["ProjectVersionArn"],
                        )
                        found_tags.append(
                            {
                                "Project": project["ProjectArn"],
                                "ModelVersion": model["ProjectVersionArn"],
                            }
                        )

        if found is False:
            logger.info("No match for Tag %s with value %s.", key, value)
        return found_tags
    except ClientError as err:
        logger.info("Problem finding tags: %s. ", format(err))
```

```
        raise


def main():
    """
    Entry point for example.
    """
    logging.basicConfig(level=logging.INFO,
                        format="%(levelname)s: %(message)s")

    # Set up command line arguments.
    parser = argparse.ArgumentParser(usage=argparse.SUPPRESS)

    parser.add_argument("tag", help="The tag that you want to find.")
    parser.add_argument("value", help="The tag value that you want to find.")

    args = parser.parse_args()
    key = args.tag
    value = args.value

    print(f"Searching your models for tag: {key} with value: {value}.")


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

    # Get tagged models for all projects.
    tagged_models = find_tag_in_projects(rekognition_client, key, value)

    print("Matched models\n--------------")
    if len(tagged_models) > 0:
        for model in tagged_models:
            print(
                "Project: {project}\nModel version: {version}\n".format(
                    project=model["Project"], version=model["ModelVersion"]
                )
            )

    else:
        print("No matches found.")

    print("Done.")
```

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

2. En el símbolo del sistema, escriba lo siguiente. Sustituya *key* y *value* por el nombre de la clave y el valor de la clave que desee buscar.

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

Eliminación de etiquetas de un modelo

Para eliminar una o más etiquetas de un modelo, utilice la UntagResourceoperación. Indique el ARN del modelo (`ResourceArn`) y las claves de etiqueta (`Tag-Keys`) que desee eliminar.

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

Si lo prefiere, también puede indicar `tag-keys` en este formato.

```
--tag-keys key1,key2
```

Descripción de un modelo (SDK)

Puede utilizar la API de `DescribeProjectVersions` para obtener información sobre la versión de un modelo. Si no indica `VersionName`, `DescribeProjectVersions` devolverá las descripciones de todas las versiones del modelo en el proyecto.

Cómo describir un modelo (SDK)

1. Si aún no lo ha hecho, instale y configure el AWS CLI y el AWS SDKs. Para obtener más información, consulte Paso 4: Configure y AWS CLIAWS SDKs.
2. Incluya la descripción de la versión de un modelo con el siguiente código.

   AWS CLI

   Cambie el valor de `project-arn` por el ARN del proyecto que desee describir. Cambie el valor de `version-name` por la versión del modelo que desee describir.

   ```
   aws rekognition describe-project-versions --project-arn project_arn \
   ```

```
  --version-names version_name \
  --profile custom-labels-access
```

Python

Utilice el siguiente código. Indique los siguientes parámetros de línea de comandos:

- project_arn: el ARN del modelo que desea describir.
- model_version: la versión del modelo que desea describir.

Por ejemplo: `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

Utilice el siguiente código. Indique los siguientes parámetros de línea de comandos:

- project_arn: el ARN del modelo que desea describir.
- model_version: la versión del modelo que desea describir.

```
/*
   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);
        }

    }

}
```

Copia de un modelo de Etiquetas personalizadas de Amazon Rekognition (SDK)

Puede utilizar la CopyProjectVersionoperación para copiar una versión del modelo Amazon Rekognition Custom Labels de un proyecto de Amazon Rekognition Custom Labels de origen a un proyecto de destino. El proyecto de destino puede estar en una AWS cuenta diferente o en la misma cuenta. AWS Un escenario típico consiste en copiar un modelo probado de una AWS cuenta de desarrollo a una AWS cuenta de producción.

Si lo prefiere, puede entrenar el modelo en la cuenta de destino con el conjunto de datos de origen. El uso de la operación `CopyProjectVersion` tiene las siguientes ventajas.

- La respuesta del modelo es uniforme. El entrenamiento del modelo no es determinista y no se garantiza que dos modelos entrenados con el mismo conjunto de datos hagan las mismas predicciones. Copiar el modelo con `CopyProjectVersion` permite garantizar que la respuesta del modelo copiado sea coherente con el modelo de origen y no sea necesario volver a probar el modelo.
- No es necesario el entrenamiento del modelo. Esto le permitirá ahorrar dinero, ya que se le cobrará por cada entrenamiento de un modelo que salga bien.

Para copiar un modelo a otra AWS cuenta, debes tener un proyecto de Amazon Rekognition Custom Labels en la cuenta de destino. AWS Para obtener información sobre cómo crear un proyecto, consulte Creación de un proyecto. Asegúrese de crear el proyecto en la cuenta de destino. AWS

Una política de proyecto es una política basada en recursos que establece los permisos de copia para la versión del modelo que desee copiar. Deberás usar una política de proyecto cuando el proyecto de destino esté en una AWS cuenta diferente a la del proyecto de origen.

No es necesario utilizar una política de proyecto para copiar versiones de modelos en la misma cuenta. Sin embargo, puede utilizar una política de proyecto en proyectos entre cuentas si desea tener más control sobre estos recursos.

Para adjuntar la política del proyecto al proyecto de origen, debe llamar a la PutProjectPolicyoperación.

No se puede utilizar `CopyProjectVersion` para copiar un modelo a un proyecto de una AWS región diferente. Además, no es posible crear un modelo a través de la consola de Etiquetas personalizadas de Amazon Rekognition. En estos casos, puede entrenar el modelo en el proyecto de destino con los conjuntos de datos utilizados para entrenar el modelo de origen. Para obtener más información, consulte Entrenamiento de un modelo de Etiquetas personalizadas de Amazon Rekognition.

Para copiar un modelo de un proyecto de origen en un proyecto de destino, haga lo siguiente:

Cómo copiar un modelo

1. Cree un documento de política de proyecto.
2. Adjunte la política del proyecto al proyecto de origen.
3. Copie el modelo con la operación `CopyProjectVersion`.

Para eliminar una política de proyecto de un proyecto, llama DeleteProjectPolicy. Para obtener una lista de las políticas de proyecto adjuntas a un proyecto, llama ListProjectPolicies.

Temas

- Creación de un documento de política del proyecto
- Asociación de una política de proyecto (SDK)
- Copia de un modelo (SDK)
- Cómo ver las políticas de proyecto (SDK)
- Eliminación de una política de proyecto (SDK)

Creación de un documento de política del proyecto

Etiquetas personalizadas de Amazon Rekognition utiliza una política basada en recursos, conocida como política de proyecto, destinada a gestionar los permisos de copia de la versión de un modelo. Una política de proyecto es un documento en formato JSON.

La política de proyecto permite o deniega el permiso a una entidad principal para que copie la versión de un modelo de un proyecto de origen en un proyecto de destino. Necesitas una política de proyecto si el proyecto de destino está en una AWS cuenta diferente. Esto también se aplica si el proyecto de destino está en la misma cuenta de AWS que el proyecto de origen y se quiere restringir el acceso a versiones específicas del modelo. Por ejemplo, es posible que desee denegar los permisos de copia a un rol de IAM específico dentro de una AWS cuenta.

En el siguiente ejemplo, se permite a la entidad principal `arn:aws:iam::111111111111:role/Admin` para que copie la versión del modelo `arn:aws:rekognition:us-east-1:123456789012:project/my_project/version/test_1/1627045542080`.

```
{
  "Version":"2012-10-17",
  "Statement":[
    {
      "Effect":"Allow",
      "Principal":{
        "AWS":"arn:aws:iam::111111111111:role/Admin"
      },
      "Action":"rekognition:CopyProjectVersion",
      "Resource":"arn:aws:rekognition:us-east-1:111111111111:project/my_project/
version/test_1/1627045542080"
    }
  ]
}
```

> **Note**
>
> En un documento de política de proyecto, los campos `Action`, `Resource`, `Principal` y `Effect` son obligatorios.
> La única `action` que se admite es `rekognition:CopyProjectVersion`.
> Los campos `NotAction`, `NotResource` y `NotPrincipal` están prohibidos y no deben estar presentes en el documento de política de proyecto.

Si no especificas una política de proyecto, un director de la misma AWS cuenta que el proyecto de origen puede seguir copiando un modelo si el director tiene una política basada en la identidad, por ejemplo `AmazonRekognitionCustomLabelsFullAccess`, que dé permiso para llamar. `CopyProjectVersion`

En el siguiente procedimiento se crea un archivo de documento de política de proyecto que puede utilizar con el ejemplo de Python en Asociación de una política de proyecto (SDK). Si usa el `put-project-policy` AWS CLI comando, proporciona la política del proyecto como una cadena JSON.

Cómo un documento de política de proyecto

1. En un editor de texto, cree el siguiente documento. Cambie los siguientes valores:

   - Effect: indique `ALLOW` para conceder el permiso de copia. Indique `DENY` para denegar el permiso de copia.
   - Principal: elija la entidad principal a la que desee permitir o denegar el acceso a las versiones del modelo que indique en `Resource`. Por ejemplo, puede especificar el principal de la cuenta de AWS para una AWS cuenta diferente. No existen limitaciones a las entidades principales que puede usar. Para obtener más información, consulte Especificación de una entidad principal.
   - Resource: el nombre de recurso de Amazon (ARN) de la versión del modelo para la que desea conceder o no los permisos de copia. Si desea conceder permisos a todas las versiones del modelo del proyecto de origen, utilice el siguiente formato `arn:aws:rekognition:`*`region`*`:`*`account`*`:project/`*`source project`*`/version/*`

```
{
  "Version":"2012-10-17",
  "Statement":[
    {
      "Effect":"ALLOW or DENY",
      "Principal":{
        "AWS":"principal"
      },
      "Action":"rekognition:CopyProjectVersion",
      "Resource":"Model version ARN"
    }
  ]
}
```

2. Guarde la política del proyecto en su ordenador.
3. Asocie la política del proyecto al proyecto de origen según las instrucciones que aparecen en Asociación de una política de proyecto (SDK).

Asociación de una política de proyecto (SDK)

Para adjuntar una política de proyecto a un proyecto de Amazon Rekognition Custom Labels, debe llamar a la operación. PutProjectpolicy

Asocie varias políticas de proyecto en un proyecto llamando a `PutProjectPolicy` por cada política de proyecto que desee agregar. Puedes asociar hasta cinco políticas de proyecto a un proyecto. Si necesita asociar más políticas de proyecto, puede solicitar que se aumente el límite.

La primera vez que asocie una política de proyecto única a un proyecto, no indique ningún ID de revisión en el parámetro de entrada `PolicyRevisionId`. La respuesta de `PutProjectPolicy` es un ID de revisión de la política de proyecto que Etiquetas personalizadas de Amazon Rekognition crea para usted. Puede usar el ID de revisión para actualizar o eliminar la última revisión de una política de proyecto. Etiquetas personalizadas de Amazon Rekognition solo conserva la última revisión de la política de un proyecto. Si intenta modificar o eliminar la revisión anterior de una política de proyecto, le aparecerá el error `InvalidPolicyRevisionIdException`.

Para modificar una política de proyecto existente, indique el ID de revisión de la política del proyecto en el parámetro de entrada `PolicyRevisionId`. Puede obtener la revisión de las políticas IDs de un proyecto llamando al teléfono. ListProjectPolicies

Después de asociar una política de proyecto a un proyecto de origen, puede copiar el modelo del proyecto de origen al proyecto de destino. Para obtener más información, consulte Copia de un modelo (SDK).

Para eliminar una política de proyecto de un proyecto, llama DeleteProjectPolicy. Para obtener una lista de las políticas de proyecto adjuntas a un proyecto, llama ListProjectPolicies.

Cómo asociar una política de proyecto a un proyecto (SDK)

1. Si aún no lo ha hecho, instale y configure el AWS CLI y el AWS SDKs. Para obtener más información, consulte Paso 4: Configure y AWS CLIAWS SDKs.
2. Cree un documento de política de proyecto.
3. Utilice el siguiente código para adjuntar la política del proyecto al proyecto, en la AWS cuenta de confianza, que contiene la versión del modelo que desea copiar. Para obtener el ARN del proyecto, llame. DescribeProjects Para obtener la versión del modelo ARN, llame. DescribeProjectVersions

AWS CLI

Cambie los siguientes valores:

- `project-arn`nal ARN del proyecto de origen en la AWS cuenta de confianza que contiene la versión del modelo que desea copiar.
- `policy-name` por el nombre de política que elija.
- `principal`: por la entidad principal a la que desee permitir o denegar el acceso a las versiones del modelo que indique en `Model version ARN`.
- `project-version-arn` por el ARN de la versión del modelo que desee copiar.

Si desea modificar una política de proyecto existente, indique el parámetro `policy-revision-id` e incluya el ID de revisión de la política del proyecto deseada.

```
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

Utilice el siguiente código. Indique los siguientes parámetros de línea de comandos:

- `project_arn`: el ARN del proyecto de origen al que desee asociar la política del proyecto.
- `policy_name`: el nombre de política que elija.
- `project_policy`: el archivo que contiene el documento de la política de proyecto.
- `policy_revision_id`: (Opcional). Si desea modificar la revisión de una política de proyecto existente, indique el ID de revisión de la política de proyecto.

```
# 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

Utilice el siguiente código. Indique los siguientes parámetros de línea de comandos:

- `project_arn`: el ARN del proyecto de origen al que desee asociar la política del proyecto.
- `project_policy_name`: el nombre de política que elija.
- `project_policy_document`: el archivo que contiene el documento de la política de proyecto.
- `project_policy_revision_id`: (Opcional). Si desea modificar la revisión de una política de proyecto existente, indique el ID de revisión de la política de proyecto.

```
/*
   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. Copie la versión del modelo siguiendo las instrucciones que aparecen en Copia de un modelo (SDK).

Copia de un modelo (SDK)

Puede usar la API de `CopyProjectVersion` para copiar una versión del modelo de un proyecto de origen en un proyecto de destino. El proyecto de destino puede estar en una AWS cuenta diferente, pero debe estar en la misma AWS región. Si el proyecto de destino está en una AWS cuenta diferente (o si desea conceder permisos específicos para una versión del modelo

copiada en una AWS cuenta), debe adjuntar una política de proyecto al proyecto de origen. Para obtener más información, consulte Creación de un documento de política del proyecto. La API de `CopyProjectVersion` necesita acceder al bucket de Amazon S3.

El modelo copiado incluye los resultados de entrenamiento del modelo de origen, pero no incluye los conjuntos de datos de origen.

La AWS cuenta de origen no es propietaria del modelo copiado en una cuenta de destino, a menos que configure los permisos adecuados.

Cómo copiar un modelo (SDK)

1. Si aún no lo ha hecho, instale y configure el AWS CLI y el AWS SDKs. Para obtener más información, consulte Paso 4: Configure y AWS CLIAWS SDKs.
2. Asocie la política de proyecto al proyecto de origen siguiendo las instrucciones que aparecen en Asociación de una política de proyecto (SDK).
3. Si va a copiar el modelo a una AWS cuenta diferente, asegúrese de tener un proyecto en la AWS cuenta de destino.
4. Utilice el siguiente código para copiar la versión del modelo en un proyecto de destino.

   AWS CLI

   Cambie los siguientes valores:

   - `source-project-arn` por el ARN del proyecto de origen que incluya la versión del modelo que desee copiar.
   - `source-project-version-arn` por el ARN de la versión del modelo que desee copiar.
   - `destination-project-arn` por el ARN del proyecto de destino en el que desee copiar el modelo.
   - `version-name` por el nombre de la versión del modelo en el proyecto de destino.
   - `bucket` por el bucket de S3 en el que desee copiar los resultados del entrenamiento para el modelo de origen.
   - `folder` por la carpeta en `bucket` en la que desee copiar los resultados del entrenamiento para el modelo de origen.
   - (Opcional) `kms-key-id` por el ID de la clave de AWS Key Management Service del modelo.
   - (Opcional) `key`: por la clave de etiqueta que elija.

- (Opcional) `value`: por el valor de etiqueta que elija.

```
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

Utilice el siguiente código. Indique los siguientes parámetros de línea de comandos:

- `source_project_arn`— el ARN del proyecto de origen en la AWS cuenta de origen que contiene la versión del modelo que desea copiar.
- `source_project_version-arn`— el ARN de la versión del modelo en la AWS cuenta de origen que desea copiar.
- `destination_project_arn`: por el ARN del proyecto de destino en el que desee copiar el modelo.
- `destination_version_name`: por el nombre de la versión del modelo en el proyecto de destino.
- `training_results`: por la ubicación de S3 en la que desee copiar los resultados del entrenamiento para la versión del modelo de origen.
- (Opcional) `kms_key_id`: por el ID de la clave de AWS Key Management Service del modelo.
- (Opcional) `tag_name`: por la clave de etiqueta que elija.
- (Opcional) `tag_value`: por el valor de etiqueta que elija.

```
# 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

Utilice el siguiente código. Indique los siguientes parámetros de línea de comandos:

- `source_project_arn`— el ARN del proyecto de origen en la AWS cuenta de origen que contiene la versión del modelo que desea copiar.
- `source_project_version-arn`— el ARN de la versión del modelo en la AWS cuenta de origen que desea copiar.
- `destination_project_arn`: por el ARN del proyecto de destino en el que desee copiar el modelo.
- `destination_version_name`: por el nombre de la versión del modelo en el proyecto de destino.
- `output_bucket`: por el bucket de S3 en el que desee copiar los resultados del entrenamiento para la versión del modelo de origen.
- `output_folder`: por la carpeta en S3 en la que desee copiar los resultados del entrenamiento para la versión del modelo de origen.

```
/*
   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);
        }

    }

}
```

Cómo ver las políticas de proyecto (SDK)

Puede usar la ListProjectPoliciesoperación para enumerar las políticas del proyecto que se adjuntan a un proyecto de Amazon Rekognition Custom Labels.

Cómo ver las políticas de proyecto asociadas a un proyecto (SDK)

1. Si aún no lo ha hecho, instale y configure el y el AWS CLI . AWS SDKs Para obtener más información, consulte Paso 4: Configure y AWS CLIAWS SDKs.
2. Usa el siguiente código para ver las políticas de proyecto.

   AWS CLI

   Cambie `project-arn` por el nombre del recurso de Amazon del proyecto para el que desee incluir las políticas de proyecto asociadas.

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

   Python

   Utilice el siguiente código. Indique los siguientes parámetros de línea de comandos:

   - project_arn: el nombre del recurso de Amazon del proyecto para el que desee incluir las políticas de proyecto asociadas.

   Por ejemplo: `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

Utilice el siguiente código. Indique los siguientes parámetros de línea de comandos:

- project_arn: el ARN del proyecto que incluya las políticas de proyecto que desee ver.

```
/*
   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);
        }
```

```
    }

}
```

Eliminación de una política de proyecto (SDK)

Puede utilizar la DeleteProjectPolicyoperación para eliminar una revisión de una política de proyecto existente de un proyecto de etiquetas personalizadas de Amazon Rekognition. Si desea eliminar todas las revisiones de una política de proyecto que estén adjuntas a un proyecto, utilice esta opción ListProjectPoliciespara obtener la revisión IDs de cada política de proyecto adjunta al proyecto. A continuación, llame a `DeletePolicy` por cada nombre de política.

Cómo eliminar la revisión de una política de proyecto (SDK)

1. Si aún no lo ha hecho, instale y configure el AWS CLI y el AWS SDKs. Para obtener más información, consulte Paso 4: Configure y AWS CLIAWS SDKs.
2. Utilice el código siguiente para eliminar una política de proyecto.

   DeletePolicy toma`ProjectARN`, `PolicyName` y`PolicyRevisionId`. `ProjectARN`y `PolicyName` son necesarios para esta API. `PolicyRevisionId`es opcional, pero se puede incluir para las actualizaciones atómicas.

   AWS CLI

   Cambie los siguientes valores:

   - `policy-name` por el nombre de la política de proyecto que desee eliminar.
   - `policy-revision-id` por el ID de revisión de la política de proyecto que desee eliminar.
   - `project-arn` por el nombre del recurso de Amazon del proyecto que incluya la revisión de la política de proyecto que desee eliminar.

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

Python

Utilice el siguiente código. Indique los siguientes parámetros de línea de comandos:

- `policy-name`: el nombre de la política de proyecto que desee eliminar.
- `project-arn`: el nombre del recurso de Amazon del proyecto que incluya la política de proyecto que desee eliminar.
- `policy-revision-id`: el ID de revisión de la política de proyecto que desee eliminar.

Por ejemplo: `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

Utilice el siguiente código. Indique los siguientes parámetros de línea de comandos:

- `policy-name`: el nombre de la política de proyecto que desee eliminar.
- `project-arn`: el nombre del recurso de Amazon del proyecto que incluya la política de proyecto que desee eliminar.
- `policy-revision-id`: el ID de revisión de la política de proyecto que desee eliminar.

```
/*
   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);
        }

    }

}
```

Ejemplos de etiquetas personalizadas

Esta sección contiene ejemplos que muestran cómo utilizar las capacidades de Etiquetas personalizadas de Amazon Rekognition.

Ejemplo	Descripción
Mejora de un modelo con Model feedback	Indica cómo mejorar un modelo mediante la verificación humana para crear un nuevo conjunto de datos de entrenamiento.
Demostración de Etiquetas personalizadas de Amazon Rekognition	Demostración de una interfaz de usuario donde salen los resultados de una llamada a `DetectCustomLabels` .
Detección de etiquetas personalizadas en vídeos	Indica cómo se puede utilizar `DetectCustomLabels` con fotogramas extraídos de un vídeo.
Análisis de imágenes con una AWS Lambda función	Indica cómo puede utilizar `DetectCustomLabels` con una función de Lambda.
Creación de un archivo de manifiesto a partir de un archivo CSV	Indica cómo usar un archivo CSV para crear un archivo de manifiesto adecuado para buscar objetos, escenas y conceptos asociados a una imagen completa (clasificación).

Mejora de un modelo con Model feedback

La solución Model Feedback le permite dar su opinión sobre las predicciones del modelo y realizar mejoras mediante la verificación humana. Según cada caso concreto, puede tener mejor resultados con un conjunto de datos de entrenamiento que tenga solo unas pocas imágenes. Es posible que se necesite un conjunto de entrenamiento anotado más grande para crear un modelo más preciso. Con la solución Model Feedback, puede crear un conjunto de datos más grande mediante la asistencia del modelo.

Para instalar y configurar la solución Model Feedback, consulte Solución Model Feedback.

El proceso de trabajo para seguir mejorando el modelo es el siguiente:

1. Entrene la primera versión del modelo (posiblemente con un conjunto de datos de entrenamiento pequeño).
2. Use un conjunto de datos sin anotaciones para la solución Model Feedback.
3. La solución Model Feedback utiliza el modelo actual. Ejecuta tareas de verificación humana para anotar un nuevo conjunto de datos.
4. En función de los comentarios de las personas, la solución Model Feedback genera un archivo de manifiesto que se utiliza para crear un nuevo modelo.

Demostración de Etiquetas personalizadas de Amazon Rekognition

La demostración de etiquetas personalizadas de Amazon Rekognition muestra una interfaz de usuario que analiza las imágenes de su ordenador local mediante la API. DetectCustomLabels

La aplicación te muestra información sobre los modelos de Amazon Rekognition Custom Labels de tu cuenta. AWS Después de seleccionar un modelo en ejecución, puede analizar una imagen a través de su ordenador local. Si es necesario, puede iniciar un modelo. También puede detener un modelo en ejecución. La aplicación muestra la integración con otros servicios de AWS, como Amazon Cognito, Amazon S3 y Amazon. CloudFront

Para obtener más información, consulte Demostración de Etiquetas personalizadas de Amazon Rekognition.

Detección de etiquetas personalizadas en vídeos

En el siguiente ejemplo, se indica cómo se puede utilizar `DetectCustomLabels` con fotogramas extraídos de un vídeo. El código se ha probado con archivos de vídeo en formato mov y mp4.

Utilización de **`DetectCustomLabels`** con fotogramas capturados

1. Si aún no lo ha hecho, instale y configure el AWS CLI y el AWS SDKs. Para obtener más información, consulte Paso 4: Configure y AWS CLIAWS SDKs.
2. Asegúrese de que tiene los permisos `rekognition:DetectCustomLabels` y `AmazonS3ReadOnlyAccess`. Para obtener más información, consulte Paso 4: Configure y AWS CLIAWS SDKs.

3. Use el siguiente código de ejemplo. Cambie el valor de `videoFile` por el nombre de archivo de vídeo. Cambie el valor de `projectVersionArn` por el nombre de recurso de Amazon (ARN) del modelo de Etiquetas personalizadas de Amazon Rekognition.

```
# 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()
```

Análisis de imágenes con una AWS Lambda función

AWS Lambda es un servicio informático que le permite ejecutar código sin aprovisionar ni administrar servidores. Por ejemplo, puede analizar las imágenes enviadas desde una aplicación móvil sin tener que crear un servidor para alojar el código de la aplicación. En las instrucciones siguientes se explica cómo crear una función de Lambda en Python que llame a DetectCustomLabels. La función analiza una imagen facilitada y devuelve una lista con las etiquetas que se encuentran en la imagen. Las instrucciones incluyen un ejemplo de código Python que indica cómo llamar a la función de Lambda con la imagen de un bucket de Amazon S3 o una imagen sacada de un ordenador local.

Temas

- Paso 1: Crear una AWS Lambda función (consola)
- Paso 2: (opcional) Crear una capa (consola)
- Paso 3: Añadir código de Python (consola)
- Paso 4: Probar la función de Lambda

Paso 1: Crear una AWS Lambda función (consola)

En este paso, se crea una AWS función vacía y una función de ejecución de IAM que permite a la función llamar a la `DetectCustomLabels` operación. También da acceso al bucket de Amazon

S3 que almacena imágenes para su análisis. También puede indicar variables de entorno para lo siguiente:

- El modelo de Etiquetas personalizadas de Amazon Rekognition para el que desea usar la función de Lambda.
- El límite de confianza que desea que utilice el modelo.

Más adelante, añada el código fuente y, si lo desea, una capa a la función de Lambda.

Para crear una AWS Lambda función (consola)

1. Inicie sesión en AWS Management Console y abra la AWS Lambda consola en https://console.aws.amazon.com/lambda/.
2. Seleccione Crear función. Para obtener más información, consulte Crear una función de Lambda con la consola.
3. Elija las siguientes opciones.
   - Elija Crear desde cero.
   - Introduzca un valor en Nombre de función.
   - En Tiempo de ejecución, elija Python 3.10.
4. Elija Crear función para crear la función AWS Lambda .
5. En la página de la función, seleccione la pestaña Configuración.
6. En el panel Variables de entorno, elija Editar.
7. Añada las siguientes variables de entorno. En cada variable, elija Añadir variable de entorno y luego introduzca la clave y el valor de la variable.

Clave	Valor
MODEL_ARN	El nombre de recurso de Amazon (ARN) del modelo que desea que utilice la función de Lambda. Puede obtener el ARN del modelo en la pestaña Usar modelo que hay en la página de detalles del modelo en la consola de Etiquetas personalizadas de Amazon Rekognition.

Clave	Valor
CONFIDENCE	El valor mínimo (0-100) de confianza del modelo en la predicción de una etiqueta. La función de Lambda no devuelve etiquetas con valores de confianza inferiores a este valor.

8. Elija Guardar para guardar las variables de entorno.
9. En el panel Permisos, en Nombre del rol, elija el rol de ejecución para abrir el rol en la consola de IAM.
10. En la pestaña Permisos, elija Agregar permisos y después Crear política insertada.
11. Elija JSON y reemplace la política existente por la siguiente política.

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

12. Elija Next (Siguiente).
13. En Detalles de la política, introduce un nombre para la política, como DetectCustomLabels-access.
14. Elija Crear política.
15. Si va a almacenar imágenes para analizarlas en un bucket de Amazon S3, repita los pasos del 10 al 14.

    a. En el paso 11, utilice la siguiente política. *bucket/folder path*Sustitúyala por la ruta del depósito y la carpeta de Amazon S3 a las imágenes que desee analizar.

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

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

b. En el paso 13, elija un nombre de política diferente, como S3Bucket-access.

Paso 2: (opcional) Crear una capa (consola)

Para ejecutar este ejemplo, no es necesario realizar este paso. La `DetectCustomLabels` operación se incluye en el entorno Lambda Python predeterminado como parte del AWS SDK para Python (Boto3). Si otras partes de la función Lambda necesitan actualizaciones de AWS servicio recientes que no estén en el entorno Lambda Python predeterminado, realice este paso para añadir la última versión del SDK de Boto3 como capa a su función.

En primer lugar, se crea un archivo de archivo .zip con el SDK de Boto3. A continuación, cree una capa y añada el archivo de zip a la capa. Para obtener más información, consulte Uso de capas con la función de Lambda.

Cómo crear y añadir una capa (consola)

1. Abra el símbolo del sistema y escriba el comando siguiente.

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

2. Apunte el nombre del archivo zip (boto3-layer.zip). Lo necesitará más tarde en el paso 6 de este mismo procedimiento.
3. Abra la consola en. AWS Lambda https://console.aws.amazon.com/lambda/
4. En el panel de navegación, elija Capas.
5. Elija Crear capa.
6. Introduzca los valores correspondientes en Nombre y Descripción.
7. Elija Cargar archivo .zip y luego Cargar.

8. En el cuadro de diálogo, elija el archivo .zip (boto3-layer.zip) que creó en el paso 1 de este procedimiento.
9. Con tiempos de ejecución compatibles, elija Python 3.9.
10. Elija Crear para crear la capa.
11. Elija el icono del menú del panel de navegación.
12. Seleccione Funciones en el panel de navegación.
13. En la lista de recursos, elija la función que haya creado en Paso 1: Crear una AWS Lambda función (consola).
14. Elija la pestaña Código.
15. En la sección Capas, elija Agregar una capa.
16. Elija Capas personalizadas.
17. En Capas personalizadas, elija el nombre de la capa que escogió en el paso 6.
18. En Versión, elija la versión de la capa, que debe ser 1.
19. Elija Agregar.

Paso 3: Añadir código de Python (consola)

En este paso, agregue código Python a la función de Lambda mediante el editor de código de la consola de Lambda. El código analiza una imagen facilitada con `DetectCustomLabels` y devuelve una lista con las etiquetas que se encuentran en la imagen. La imagen facilitada puede estar ubicada en un bucket de Amazon S3 o incluirse como bytes de imagen codificados en byte64.

Cómo añadir código Python (consola)

1. Si no se encuentra en la consola de Lambda, haga lo siguiente:
   a. Abra la AWS Lambda consola en https://console.aws.amazon.com/lambda/.
   b. Abra la función de Lambda que creó en Paso 1: Crear una AWS Lambda función (consola).
2. Elija la pestaña Código.
3. En Código fuente, sustituya el código en lambda_function.py por lo siguiente:

```
# 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. Elija Implementar para implementar la función de Lambda.

Paso 4: Probar la función de Lambda

En este paso, utilice el código Python de su equipo para pasar una imagen local, o la imagen de un bucket de Amazon S3, a la función de Lambda. Las imágenes transferidas desde un equipo local deben tener un tamaño inferior a 6 291 456 bytes. Si las imágenes son más grandes, cárguelas en un bucket de Amazon S3 y llame al script con la ruta de Amazon S3 de la imagen. Para obtener más información sobre cómo cargar archivos en un bucket de Amazon S3, consulte Cargar objetos.

Asegúrese de ejecutar el código en la misma AWS región en la que creó la función Lambda. Puede ver la AWS región de la función Lambda en la barra de navegación de la página de detalles de la función en la consola Lambda.

Si la AWS Lambda función devuelve un error de tiempo de espera, amplíe el período de tiempo de espera de la función Lambda. Para obtener más información, consulte Configuración del tiempo de espera de la función (consola).

Para obtener más información sobre cómo invocar una función Lambda desde el código, consulte AWS Lambda Invocar funciones.

Cómo probar la función de Lambda

1. Asegúrese de que tiene el permiso `lambda:InvokeFunction`. Puede utilizar la siguiente política.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "InvokeLambda",
            "Effect": "Allow",
            "Action": "lambda:InvokeFunction",
            "Resource": "ARN for lambda function"
```

```
        }
    ]
}
```

Puede obtener el ARN de la función de Lambda en la descripción general de la función en la consola de Lambda.

Para dar acceso, asigne los permisos a los usuarios, grupos o roles correspondientes:

- Usuarios y grupos en: AWS IAM Identity Center

  Cree un conjunto de permisos. Siga las instrucciones de Creación de un conjunto de permisos en la Guía del usuario de AWS IAM Identity Center .
- Usuarios gestionados en IAM a través de un proveedor de identidades:

  Cree un rol para la federación de identidades. Siga las instrucciones descritas en Creación de un rol para un proveedor de identidad de terceros (federación) en la Guía del usuario de IAM.
- Usuarios de IAM:
  - Cree un rol que el usuario pueda aceptar. Siga las instrucciones descritas en Creación de un rol para un usuario de IAM en la Guía del usuario de IAM.
  - (No recomendado) Adjunte una política directamente a un usuario o añada un usuario a un grupo de usuarios. Siga las instrucciones descritas en Adición de permisos a un usuario (consola) de la Guía del usuario de IAM.

2. Instale y configure el AWS SDK para Python. Para obtener más información, consulte Paso 4: Configure y AWS CLIAWS SDKs.
3. Inicie el modelo que eligió en el paso 7 de Paso 1: Crear una AWS Lambda función (consola).
4. Guarde el siguiente código en un archivo denominado `client.py`.

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

"""
Purpose
Test code for running the Amazon Rekognition Custom Labels Lambda
function example code.
"""

import argparse
import logging
```

```
import base64
import json
import boto3

from botocore.exceptions import ClientError


logger = logging.getLogger(__name__)


def analyze_image(function_name, image):
    """Analyzes an image with an AWS Lambda function.
    :param image: The image that you want to analyze.
    :return The status and classification result for
    the image analysis.
    """

    lambda_client = boto3.client('lambda')

    lambda_payload = {}

    if image.startswith('s3://'):
        logger.info("Analyzing image from S3 bucket: %s", image)
        bucket, key = image.replace("s3://", "").split("/", 1)
        s3_object = {
            'Bucket': bucket,
            'Name': key
        }
        lambda_payload = {"S3Object": s3_object}

    # Call the lambda function with the image.
    else:
        with open(image, 'rb') as image_file:
            logger.info("Analyzing local image image: %s ", image)
            image_bytes = image_file.read()
            data = base64.b64encode(image_bytes).decode("utf8")

            lambda_payload = {"image": data}

    response = lambda_client.invoke(FunctionName=function_name,
                                    Payload=json.dumps(lambda_payload))

    return json.loads(response['Payload'].read().decode())
```

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

    parser.add_argument(
        "function", help="The name of the AWS Lambda function that you want " \
        "to use to analyze the image.")
    parser.add_argument(
        "image", help="The local image that you want to analyze.")


def main():
    """
    Entrypoint for script.
    """
    try:
        logging.basicConfig(level=logging.INFO,
                            format="%(levelname)s: %(message)s")

        # Get command line arguments.
        parser = argparse.ArgumentParser(usage=argparse.SUPPRESS)
        add_arguments(parser)
        args = parser.parse_args()

        # Get analysis results.
        result = analyze_image(args.function, args.image)
        status = result['statusCode']

        if status == 200:
            labels = result['body']
            labels = json.loads(labels)
            print(f"There are {len(labels)} labels in the image.")
            for custom_label in labels:
                confidence = int(round(custom_label['Confidence'], 0))
                print(
                    f"Label: {custom_label['Name']}: Confidence: {confidence}%")
        else:
            print(f"Error: {result['statusCode']}")
            print(f"Message: {result['body']}")

    except ClientError as error:
```

```
        logging.error(error)
        print(error)


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

5. Ejecute el código. En el argumento de la línea de comandos, indique el nombre de la función de Lambda y la imagen que desee analizar. Puede poner la ruta de una imagen local o la ruta S3 de una imagen almacenada en un bucket de Amazon S3. Por ejemplo:

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

   Si la imagen está en un bucket de Amazon S3, asegúrese de que es el mismo bucket que indicó en el paso 15 de Paso 1: Crear una AWS Lambda función (consola).

   Si se ejecuta correctamente, le saldrá una lista de etiquetas que se encuentran en la imagen. Si no devuelve ninguna etiqueta, considere la posibilidad de reducir el valor de confianza que eligió en el paso 7 de Paso 1: Crear una AWS Lambda función (consola).

6. Si ha terminado con la función de Lambda y otras aplicaciones no utilizan el modelo, detenga el modelo. Recuerde iniciar el modelo la próxima vez que desee utilizar la función de Lambda.

Seguridad

Puede proteger la gestión de sus proyectos, modelos y la operación `DetectCustomLabels` que utilicen sus clientes para detectar etiquetas personalizadas.

Para obtener más información sobre cómo proteger Amazon Rekognition, consulte Seguridad de Amazon Rekognition.

Protección de proyectos de Etiquetas personalizadas de Amazon Rekognition

Puede proteger sus proyectos de Etiquetas personalizadas de Amazon Rekognition si aplica los permisos de recursos que determinan las políticas basadas en identidad. Para obtener más información, consulte Políticas basadas en identidad y políticas basadas en recursos.

Los recursos de Etiquetas personalizadas de Amazon Rekognition que puede proteger son:

Recurso	Formato de nombre de recurso de Amazon
Proyecto	arn:aws:rekognition: *:project/ /datetime *project_name*
Modelo	arn:aws:rekognition: *project_name* *:project / /version/ /datetime *name*

En la siguiente política de ejemplo se indica cómo dar un permiso de identidad para:

- Describir todos los proyectos.
- Crear, iniciar, detener y utilizar un modelo específico para la inferencia.
- Crear un proyecto. Crear y describir un modelo específico.
- Denegar la creación de un proyecto específico.

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

Asegurando DetectCustomLabels

La identidad utilizada para detectar etiquetas personalizadas puede ser diferente de la identidad que administra los modelos de Etiquetas personalizadas de Amazon Rekognition.

Puede proteger el acceso de una identidad a `DetectCustomLabels` aplicando una política a la identidad. En el siguiente ejemplo se restringe el acceso a `DetectCustomLabels` únicamente a un modelo específico. La identidad no tiene acceso a ninguna de las demás operaciones de Amazon Rekognition.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "rekognition:DetectCustomLabels"


            ],
            "Resource": "arn:aws:rekognition:*:*:project/MyProject/version/MyVersion/*"
        }
    ]
}
```

Políticas administradas de AWS

Proporcionamos la política `AmazonRekognitionCustomLabelsFullAccess` AWS gestionada que puede utilizar para controlar el acceso a las etiquetas personalizadas de Amazon Rekognition. Para obtener más información, consulte la política administrada de AWS: AmazonRekognitionCustomLabelsFullAccess.

Directrices y cuotas en Etiquetas personalizadas de Amazon Rekognition

En las siguientes secciones se indican las directrices y cuotas de Etiquetas personalizadas de Amazon Rekognition.

Regiones compatibles

Para obtener una lista de AWS las regiones en las que están disponibles las etiquetas personalizadas de Amazon Rekognition, consulte Regiones y puntos de enlace de AWS en la Referencia general de Amazon Web Services.

Cuotas

En la siguiente lista figuran los límites en Etiquetas personalizadas de Amazon Rekognition. Para obtener información sobre los límites que pueden cambiarse, consulte Límites del servicio de AWS. Para cambiar un límite, consulte Creación de un caso.

Formación

- Los formatos de archivo admitidos son los formatos de imagen PNG y JPEG.
- El número máximo de conjuntos de datos de entrenamiento en la versión de un modelo es 1.
- El tamaño máximo del archivo de manifiesto del conjunto de datos es 1 GB.
- El número mínimo de etiquetas únicas por conjunto de datos de objetos, escenas y conceptos (clasificación) es 2.
- El número mínimo de etiquetas únicas por conjunto de datos de ubicación de objetos (detección) es 1.
- El número máximo de etiquetas únicas por manifiesto es 250.
- El número mínimo de imágenes por etiqueta es 1.
- El número máximo de imágenes por conjunto de datos de ubicación de objetos (detección) es 250 000.

  El límite para las AWS regiones de Asia Pacífico (Bombay) y Europa (Londres) es de 28 000 imágenes.

- El número máximo de imágenes por conjunto de datos de objetos, escenas y conceptos (clasificación) es 500 000. El valor predeterminado es 250 000. Para solicitar un aumento, consulte Creación de un caso.

  El límite para las AWS regiones de Asia Pacífico (Bombay) y Europa (Londres) es de 28 000 imágenes. No puede solicitar un aumento del límite.
- El número máximo de capas por imagen es 50.
- El número mínimo de cuadros delimitadores en una imagen es 0.
- El número máximo de cuadros delimitadores en una imagen es 50.
- La dimensión mínima de imagen del archivo de imagen en un bucket de Amazon S3 es 64 píxeles x 64 píxeles.
- La dimensión máxima de imagen del archivo de imagen en un bucket de Amazon S3 es 4096 píxeles x 4096 píxeles.
- El tamaño máximo de archivo de una imagen en un bucket de Amazon S3 es 15 MB.
- La relación de aspecto máxima de la imagen es 20:1.

Testeo

- El número máximo de conjuntos de datos de prueba en la versión de un modelo es 1.
- El tamaño máximo del archivo de manifiesto del conjunto de datos es 1 GB.
- El número mínimo de etiquetas únicas por conjunto de datos de objetos, escenas y conceptos (clasificación) es 2.
- El número mínimo de etiquetas únicas por conjunto de datos de ubicación de objetos (detección) es 1.
- El número máximo de etiquetas únicas por conjunto de datos es 250.
- El número mínimo de imágenes por etiqueta es 0.
- El número máximo de imágenes por etiqueta es 1000.
- El número máximo de imágenes por conjunto de datos con ubicación de objetos (detección) es 250 000.

  El límite para las AWS regiones de Asia Pacífico (Bombay) y Europa (Londres) es de 7.000 imágenes.

- El número máximo de imágenes por conjunto de datos de objetos, escenas y conceptos (clasificación) es 500 000. El valor predeterminado es 250 000. Para solicitar un aumento, consulte Creación de un caso.

  El límite para las AWS regiones de Asia Pacífico (Bombay) y Europa (Londres) es de 7.000 imágenes. No puede solicitar un aumento del límite.
- El número mínimo de etiquetas por imagen y manifiesto es 0.
- El número máximo de etiquetas por imagen y manifiesto es 50.
- El número mínimo de cuadros delimitadores en una imagen por manifiesto es 0.
- El número máximo de cuadros delimitadores en una imagen por manifiesto es 50.
- La dimensión mínima de imagen de un archivo de imagen en un bucket de Amazon S3 es 64 píxeles x 64 píxeles.
- La dimensión máxima de imagen de un archivo de imagen en un bucket de Amazon S3 es 4096 píxeles x 4096 píxeles.
- El tamaño máximo de archivo de una imagen en un bucket de Amazon S3 es 15 MB.
- Los formatos de archivo admitidos son los formatos de imagen PNG y JPEG.
- La relación de aspecto máxima de la imagen es 20:1.

Detección

- El tamaño máximo de las imágenes transferidas como bytes sin procesar es 4 MB.
- El tamaño máximo de archivo de una imagen en un bucket de Amazon S3 es 15 MB.
- La dimensión mínima de imagen de un archivo de imagen de entrada (almacenado en un bucket de Amazon S3 o incluido como bytes de imagen) es 64 píxeles x 64 píxeles.
- La dimensión máxima de imagen de un archivo de imagen de entrada (almacenado en un bucket de Amazon S3 o incluido como bytes de imagen) es 4096 píxeles x 4096 píxeles.
- Los formatos de archivo admitidos son los formatos de imagen PNG y JPEG.
- La relación de aspecto máxima de la imagen es 20:1.

Copia de modelos

- El número máximo de políticas de proyecto que se pueden asociar a un proyecto es 5.
- El número máximo de trabajos de copia simultáneos en un destino es 5.

Referencia de la API de Etiquetas personalizadas de Amazon Rekognition

La API de Etiquetas personalizadas de Amazon Rekognition viene incluida en la documentación como parte del contenido de referencia de la API de Amazon Rekognition. Esta es una lista de las operaciones de la API de Etiquetas personalizadas de Amazon Rekognition con enlaces al tema de referencia de la API de Amazon Rekognition correspondiente. Además, con los enlaces de referencia a las API de este documento se accede al tema de referencia correspondiente de la API de la Guía para desarrolladores de Amazon Rekognition. Para obtener más información sobre del uso de la API, consulte

En esta sección, se ofrece una descripción general del flujo de trabajo para entrenar y usar un modelo de etiquetas personalizadas de Amazon Rekognition con la consola y el SDK. AWS

> **Note**
>
> Etiquetas personalizadas de Amazon Rekognition se encarga de administrar los conjuntos de datos de un proyecto. Puede crear conjuntos de datos para sus proyectos con la consola y el SDK. AWS Si ha utilizado anteriormente Etiquetas personalizadas de Amazon Rekognition, es posible que tenga que asociar sus conjuntos de datos antiguos a un proyecto nuevo. Para obtener más información, consulte Paso 6: (Opcional) Asociar conjuntos de datos anteriores a nuevos proyectos.

Temas

Cómo decidir el tipo de modelo

Primero debe decidir qué tipo de modelo quiere entrenar, lo que depende de sus objetivos empresariales. Por ejemplo, puede entrenar un modelo para que encuentre un logotipo en las publicaciones de las redes sociales, identifique sus productos en las estanterías de las tiendas o clasifique las piezas de una máquina en una línea de montaje.

Etiquetas personalizadas de Amazon Rekognition puede entrenar los siguientes tipos de modelos:

- Buscar objetos, escenas y conceptos

- Buscar ubicaciones de objetos
- Buscar ubicación de marcas

Para ayudarle a decidir qué tipo de modelo podría entrenar, Etiquetas personalizadas de Amazon Rekognition incluye algunos ejemplos de proyectos que puede utilizar. Para obtener más información, consulte Introducción a Etiquetas personalizadas de Amazon Rekognition.

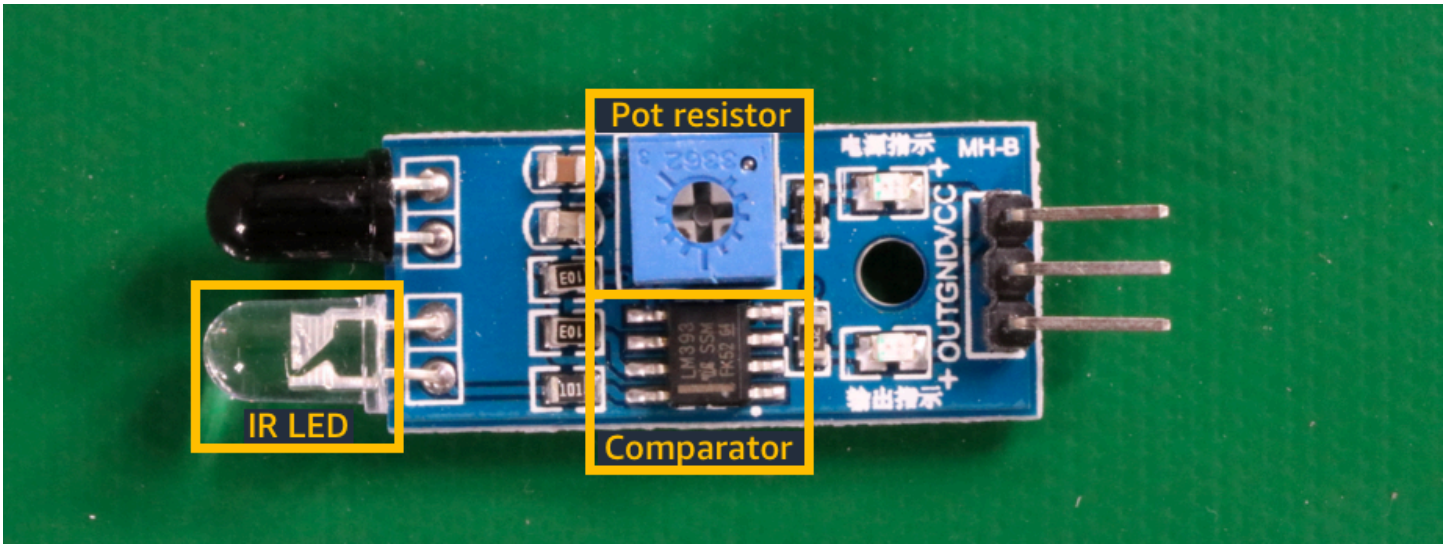
Buscar objetos, escenas y conceptos

El modelo clasifica los objetos, las escenas y los conceptos que están asociados a una imagen completa. Por ejemplo, puede entrenar un modelo para que determine si una imagen contiene una atracción turística o no. Para ver un proyecto de ejemplo, consulte Clasificación de imágenes. La siguiente imagen de un lago es un ejemplo del tipo de imagen en la que se pueden reconocer objetos, escenas y conceptos.

También tiene la opción de entrenar un modelo que clasifique las imágenes en varias categorías. Por ejemplo, la imagen anterior puede incluir categorías como el color del cielo, un reflejo o un lago. Para ver un proyecto de ejemplo, consulte Clasificación de imágenes de etiquetas múltiples.

Buscar ubicaciones de objetos

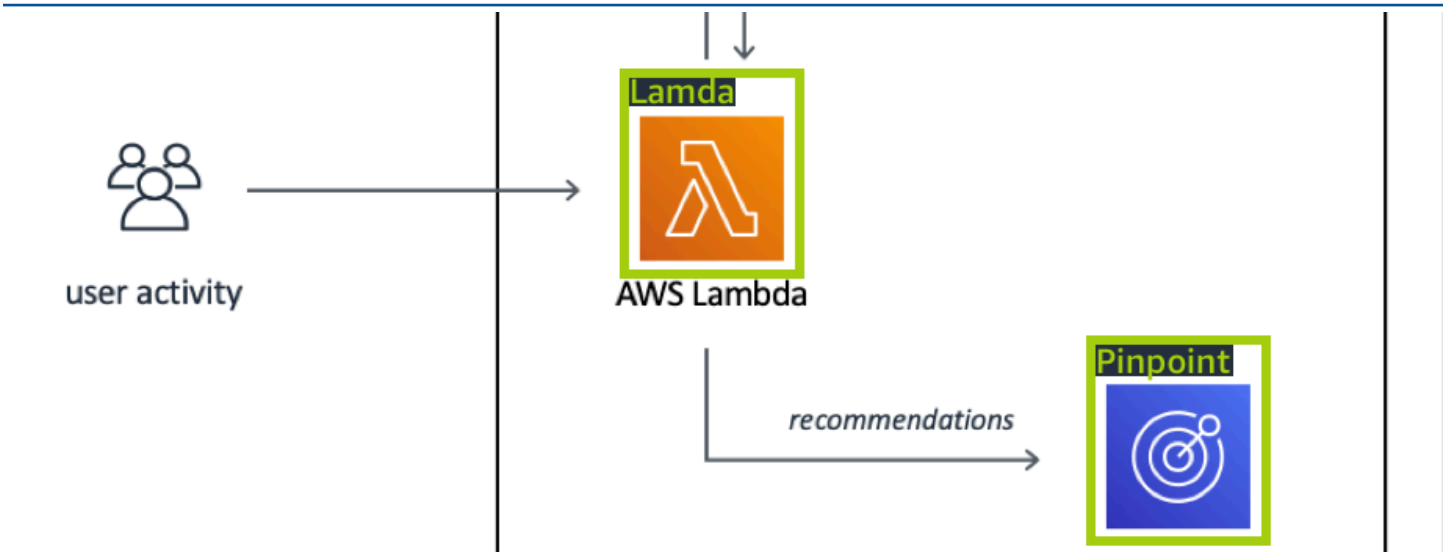
El modelo predice la ubicación de un objeto en una imagen. La predicción incluye información del cuadro delimitador en la ubicación del objeto y una etiqueta que identifica el objeto dentro del cuadro delimitador. Por ejemplo, en la siguiente imagen aparecen unos cuadros delimitadores alrededor de varios componentes de una placa de circuito, como un comparador o un potenciómetro.



En el proyecto de ejemplo Localización de objetos, se ve cómo Etiquetas personalizadas de Amazon Rekognition utiliza cuadros delimitadores etiquetados para entrenar un modelo que busca ubicaciones de objetos.

Buscar ubicación de marcas

Etiquetas personalizadas de Amazon Rekognition puede entrenar un modelo para que busque la ubicación de marcas, como los logotipos, en una imagen. La predicción incluye información del cuadro delimitador en la ubicación de la marca y una etiqueta que identifica el objeto dentro del cuadro delimitador. Para ver un proyecto de ejemplo, consulte Detección de marcas. La siguiente imagen es un ejemplo de algunas de las marcas que el modelo puede detectar.



Crear un modelo

El procedimiento para crear un modelo consiste en crear un proyecto, crear conjuntos de datos de entrenamiento y de prueba y entrenar el modelo.

Crear un proyecto

Un proyecto de Etiquetas personalizadas de Amazon Rekognition es un grupo de recursos necesarios para crear y administrar un modelo. El proyecto gestiona lo siguiente:

- Conjuntos de datos: las imágenes y las etiquetas de imagen que se utilizan para entrenar un modelo. Un proyecto incluye un conjunto de datos de entrenamiento y un conjunto de datos de prueba.

- Modelos: un modelo es un software que se entrena para buscar los conceptos, las escenas y los objetos que son exclusivos de su empresa o negocio. Puede haber varias versiones de un modelo en un proyecto.

Le recomendamos que utilice un proyecto para una sola aplicación práctica, como buscar componentes en una placa de circuito.

Puede crear un proyecto con la consola Amazon Rekognition Custom Labels y con la API. CreateProject Para obtener más información, consulte Creación de un proyecto.

Crear conjuntos de datos de entrenamiento y de prueba

Un conjunto de datos es un conjunto de imágenes y etiquetas que describen esas imágenes. Dentro del proyecto, se crea un conjunto de datos de entrenamiento y un conjunto de datos de prueba que Etiquetas personalizadas de Amazon Rekognition utiliza para entrenar y probar el modelo.

Una etiqueta identifica un objeto, una escena, un concepto o un cuadro delimitador alrededor del objeto de una imagen. Las etiquetas se asignan a una imagen completa (image-level) o se asignan a un cuadro delimitador que rodea el objeto de una imagen.

> **Important**
>
> La forma en que etiquete las imágenes en sus conjuntos de datos determina el tipo de modelo que se crea en Etiquetas personalizadas de Amazon Rekognition. Por ejemplo, para entrenar un modelo que busca objetos, escenas y conceptos, debe asignar etiquetas

de imagen a las imágenes de sus conjuntos de datos de entrenamiento y de prueba. Para obtener más información, consulte Finalidad de los conjuntos de datos.

Las imágenes deben estar en formato PNG y JPEG y se deben seguir las recomendaciones de las imágenes de entrada. Para obtener más información, consulte Preparación de imágenes.

Crear conjuntos de datos de entrenamiento y de prueba (consola)

Puede empezar con un proyecto que tenga un único conjunto de datos o un proyecto que tenga conjuntos de datos de entrenamiento y de prueba independientes. Si empieza con un único conjunto de datos, Etiquetas personalizadas de Amazon Rekognition lo divide durante el entrenamiento para crear un conjunto de datos de entrenamiento (80 %) y un conjunto de datos de prueba (20 %) para su proyecto. Comience con un único conjunto de datos si quiere que Etiquetas personalizadas de Amazon Rekognition decida qué imágenes se van a usar para el entrenamiento y las pruebas. Para tener un control total sobre el entrenamiento, las pruebas y el nivel de rendimiento, le recomendamos que inicie el proyecto con conjuntos de datos de entrenamiento y de prueba independientes.

Para crear los conjuntos de datos para un proyecto, importe las imágenes a través de uno de los siguientes métodos:

- Importar imágenes de un ordenador local.
- Importar imágenes de un bucket de S3. Etiquetas personalizadas de Amazon Rekognition puede etiquetar las imágenes con los nombres de las carpetas que contienen las imágenes.
- Importa un archivo de manifiesto de Amazon SageMaker AI Ground Truth.
- Copiar un conjunto de datos existente de Etiquetas personalizadas de Amazon Rekognition.

Para obtener más información, consulte Creación de conjuntos de datos de entrenamiento y prueba.

Según el lugar desde el que importe las imágenes, es posible que no vengan etiquetadas. Por ejemplo, las imágenes importadas de un ordenador local no están etiquetadas. Las imágenes importadas de un archivo de manifiesto de Amazon SageMaker AI Ground Truth están etiquetadas. Puede utilizar la consola de Etiquetas personalizadas de Amazon Rekognition para agregar, cambiar y asignar etiquetas. Para obtener más información, consulte Etiquetado de imágenes.

Para crear sus conjuntos de datos de entrenamiento y de prueba en la consola, consulte Creación de conjuntos de datos de entrenamiento y prueba. Si quiere ver el tutorial sobre cómo crear conjuntos de datos de entrenamiento y de prueba, consulte Clasificación de imágenes.

Crear conjuntos de datos de entrenamiento y prueba (SDK)

Puede crear sus conjuntos de datos de entrenamiento y de prueba, use la API de `CreateDataset`. Puede crear un conjunto de datos mediante un archivo de manifiesto en formato Amazon SageMaker o copiando un conjunto de datos de Etiquetas personalizadas de Amazon Rekognition existente. Para obtener más información, consulte Crear conjuntos de datos de entrenamiento y prueba (SDK) . Si es necesario, puede crear su propio archivo de manifiesto. Para obtener más información, consulte the section called “Creación de un archivo de manifiesto”.

Entrenar su modelo

Entrene su modelo con el conjunto de datos de entrenamiento. Se creará una nueva versión del modelo cada vez que se entrena. Durante el entrenamiento, Etiquetas personalizadas de Amazon Rekognition comprueba el rendimiento del modelo entrenado. Puede utilizar los resultados para evaluar y mejorar el modelo. El entrenamiento tarda un tiempo en realizarse. Solo se le cobrará cuando termine de entrenarse por completo un modelo. Para obtener más información, consulte Entrenamiento de un modelo de Etiquetas personalizadas de Amazon Rekognition. Si el entrenamiento del modelo presenta problemas, Etiquetas personalizadas de Amazon Rekognition le dará información sobre la depuración de errores que puede utilizar. Para obtener más información, consulte Depuración de un modelo de entrenamiento con errores.

Entrenar el modelo (consola)

Para entrenar su modelo a través de la consola, consulte Entrenamiento de un modelo (consola).

Entrenamiento de un modelo (SDK)

Puedes entrenar a un modelo de Amazon Rekognition Custom Labels llamando. CreateProjectVersion Para obtener más información, consulte Entrenamiento de un modelo (SDK).

Mejorar el modelo

Durante las pruebas, Etiquetas personalizadas de Amazon Rekognition genera métricas de evaluación que puede usar para mejorar el modelo entrenado.

Evaluar el modelo

Evalúe el rendimiento del modelo mediante las métricas de rendimiento creadas durante las pruebas. Las métricas de rendimiento, como la puntuación F1, las de precisión y las de exhaustividad, le

permiten conocer el rendimiento del modelo entrenado y decidir si es posible usarlo en la fase de producción. Para obtener más información, consulte Métricas para evaluar su modelo.

Evaluar un modelo (consola)

Para ver las métricas de rendimiento, consulte Acceso a las métricas de evaluación (consola).

Evaluar un modelo (SDK)

Para obtener las métricas de rendimiento, llama DescribeProjectVersionspara obtener los resultados de las pruebas. Para obtener más información, consulte Acceso a las métricas de evaluación de Etiquetas personalizadas de Amazon Rekognition (SDK) . Los resultados de las pruebas se representan con métricas que no están disponibles en la consola, como una matriz de confusión para los resultados de la clasificación. Los resultados de las pruebas se devuelven en los siguientes formatos:

- Puntuación F1: un valor único que refleja el rendimiento general de precisión y exhaustividad del modelo. Para obtener más información, consulte F1.

- Ubicación del archivo de resumen: el resumen de las pruebas incluye una serie de métricas de evaluación acumuladas de todo el conjunto de datos de pruebas y las métricas de cada etiqueta por separado. `DescribeProjectVersions` devuelve la ubicación del bucket de S3 y de la carpeta del archivo de resumen. Para obtener más información, consulte Acceder al archivo de resumen del modelo.
- Ubicación del resumen del manifiesto de evaluación: este resumen incluye los detalles sobre los resultados de las pruebas, como los índices de confianza y los resultados de las pruebas de clasificación binaria, como los falsos positivos. `DescribeProjectVersions` devuelve la ubicación del bucket de S3 y de la carpeta de los archivos de resumen. Para obtener más información, consulte Interpretación de la instantánea del manifiesto de evaluación.

Mejorar el modelo

Si se necesitan mejoras, puede agregar más imágenes de entrenamiento o mejorar el etiquetado de los conjuntos de datos. Para obtener más información, consulte Mejora de un modelo de Etiquetas personalizadas de Amazon Rekognition. También puede aportar sus observaciones sobre las predicciones que haga su modelo y utilizarlos para mejorarlo. Para obtener más información, consulte Mejora de un modelo con Model feedback.

Mejorar el modelo (consola)

Para agregar imágenes a un conjunto de datos, consulte Agregar más imágenes a un conjunto de datos. Para agregar o cambiar etiquetas, consulte the section called “Etiquetado de imágenes”.

Para volver a entrenar el modelo, consulte Entrenamiento de un modelo (consola).

Mejorar el modelo (SDK)

Para agregar imágenes a un conjunto de datos o cambiar el etiquetado de una imagen, usa la API de `UpdateDatasetEntries`. `UpdateDatasetEntries` actualiza o agrega líneas JSON a un archivo de manifiesto. Cada línea JSON contiene la información de una sola imagen, como las etiquetas asignadas o la información de los cuadros delimitadores. Para obtener más información, consulte Agregar más imágenes (SDK). Para ver las entradas de un conjunto de datos, usa la API de `ListDatasetEntries`.

Para volver a entrenar el modelo, consulte Entrenamiento de un modelo (SDK).

Iniciar el modelo

Antes de poder usar el modelo, debe iniciarlo a través de la consola de Etiquetas personalizadas de Amazon Rekognition o la API de `StartProjectVersion`. Se le cobrará por la cantidad de tiempo en que se ejecute el modelo. Para obtener más información, consulte Ejecución de un modelo entrenado.

Iniciar el modelo (consola)

Para iniciar el modelo con la consola, consulte Inicio de un modelo de Etiquetas personalizadas de Amazon Rekognition (consola).

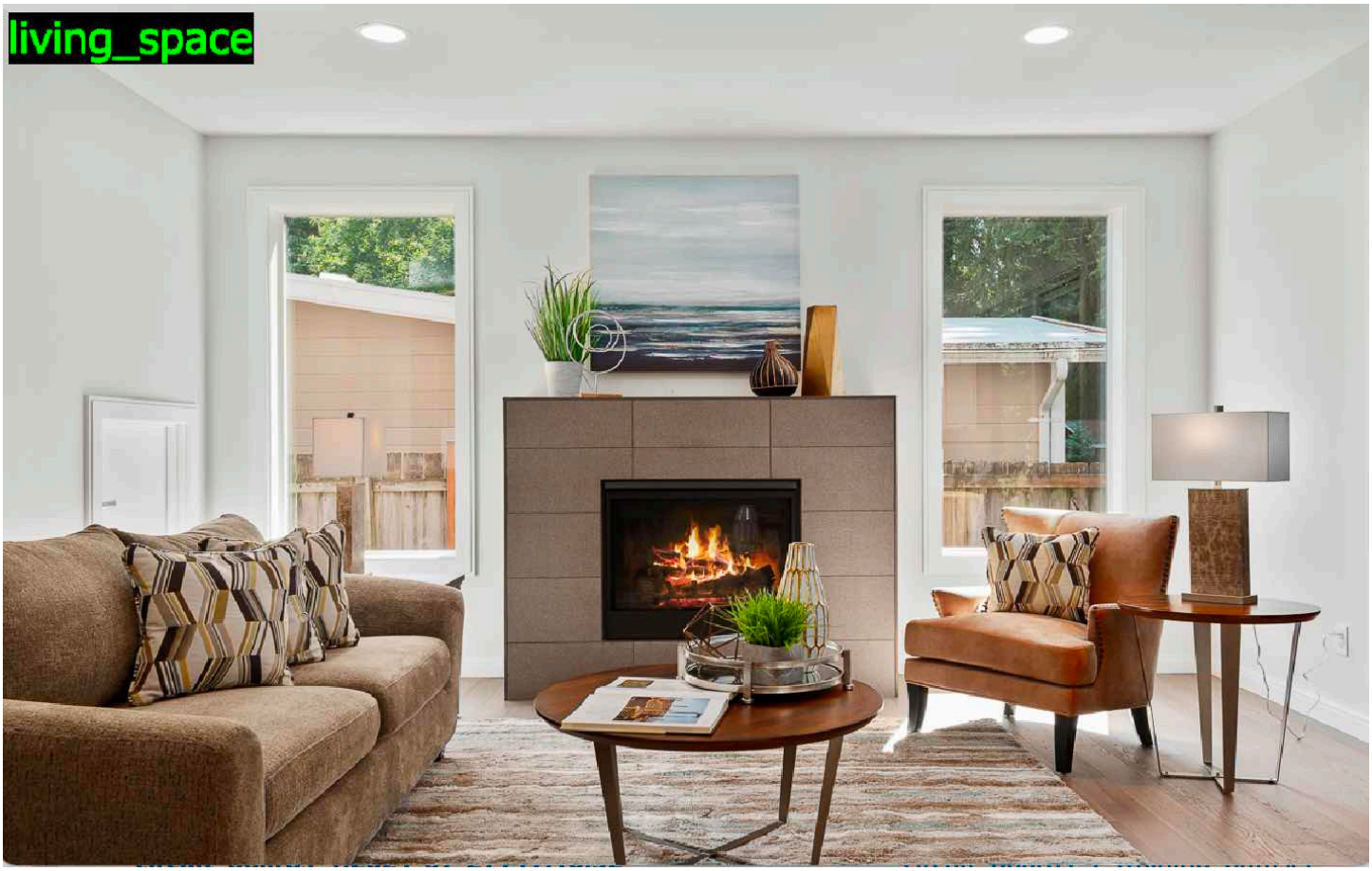
Iniciar el modelo

Empiezas a llamar a tu modelo StartProjectVersion. Para obtener más información, consulte Inicio de un modelo de Etiquetas personalizadas de Amazon Rekognition (SDK).

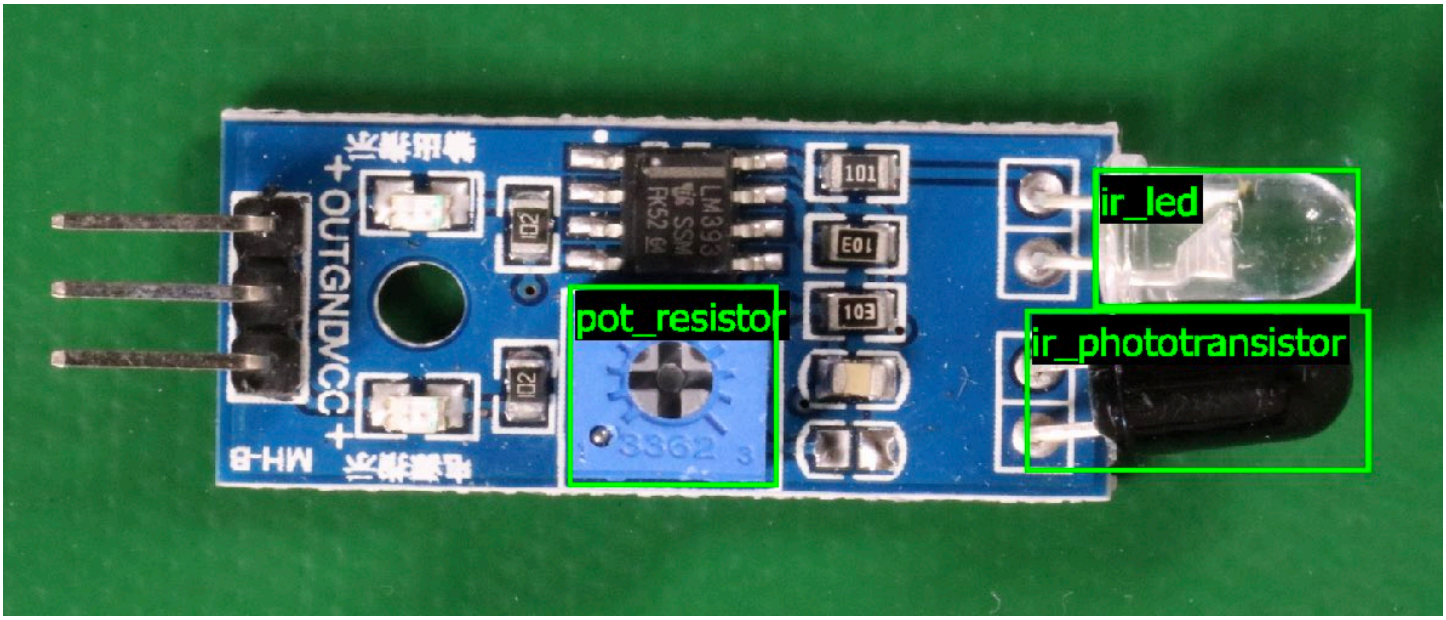
Analizar una imagen

Para analizar una imagen con el modelo, use la API de `DetectCustomLabels`. Puede indicar una imagen local o una imagen almacenada en un bucket de S3. La operación también requiere el nombre de recurso de Amazon (ARN) del modelo que desee utilizar.

Si el modelo encuentra objetos, escenas y conceptos, la respuesta incluirá una lista de etiquetas de imagen que se encuentran en la imagen. Por ejemplo, la siguiente imagen muestra las etiquetas de imagen que se encuentran en el proyecto de ejemplo Habitaciones.



Si el modelo encuentra ubicaciones de objetos, la respuesta incluirá una lista de los cuadros delimitadores etiquetados que se encuentran en la imagen. Un cuadro delimitador representa la ubicación de un objeto en una imagen. Puede utilizar la información del cuadro delimitador para dibujar un cuadro delimitador alrededor de un objeto. Por ejemplo, en la siguiente imagen aparecen unos cuadros delimitadores alrededor de los componentes de una placa de circuito que se han encontrado por medio del proyecto de ejemplo Placas de circuito.



Para obtener más información, consulte Análisis de una imagen con un modelo entrenado.

Detener el modelo

Se le cobrará por el tiempo de ejecución del modelo. Si ya no va a usar el modelo, deténgalo en la consola de Etiquetas personalizadas de Amazon Rekognition o mediante la API de `StopProjectVersion`. Para obtener más información, consulte Detención de un modelo de Etiquetas personalizadas de Amazon Rekognition.

Detener el modelo (consola)

Para detener la ejecución de un modelo con la consola, consulte Detención de un modelo de Etiquetas personalizadas de Amazon Rekognition (consola).

Detener el modelo (SDK)

Para detener a una modelo en marcha, llama StopProjectVersion. Para obtener más información, consulte Detención de un modelo de Etiquetas personalizadas de Amazon Rekognition (SDK).

.

Entrenamiento del modelo

Proyectos

- CreateProject— Crea su proyecto Amazon Rekognition Custom Labels, que es una agrupación lógica de recursos (imágenes, etiquetas, modelos) y operaciones (formación, evaluación y detección).
- DeleteProject— Elimina un proyecto de Amazon Rekognition Custom Labels.
- DescribeProjects— Devuelve una lista de todos tus proyectos de Amazon Rekognition Custom Labels.

Políticas de proyecto

- PutProjectPolicy— Adjunta una política de proyecto a un proyecto de Amazon Rekognition Custom Labels en una cuenta de confianza. AWS
- ListProjectPolicies— Devuelve una lista de las políticas de proyecto adjuntas a un proyecto.
- DeleteProjectPolicy— Elimina una política de proyecto existente.

Conjuntos de datos

- CreateDataset— Crea un conjunto de datos de etiquetas personalizadas de Amazon Rekognition.
- DeleteDataset— Elimina un conjunto de datos de etiquetas personalizadas de Amazon Rekognition.
- DescribeDataset— Describe un conjunto de datos de etiquetas personalizadas de Amazon Rekognition.
- DistributeDatasetEntries— Distribuye las entradas (imágenes) de un conjunto de datos de entrenamiento entre el conjunto de datos de entrenamiento y el conjunto de datos de prueba de un proyecto.
- ListDatasetEntries— Devuelve una lista de entradas (imágenes) de un conjunto de datos de etiquetas personalizadas de Amazon Rekognition.
- ListDatasetLabels— Devuelve una lista de etiquetas asignadas a un conjunto de datos de etiquetas personalizadas de Amazon Rekognition.
- UpdateDatasetEntries— Añade o actualiza entradas (imágenes) en un conjunto de datos de etiquetas personalizadas de Amazon Rekognition.

Modelos

- CreateProjectVersion— Entrena tu modelo de etiquetas personalizadas Amazon Rekognition.
- CopyProjectVersion— Copia tu modelo de etiquetas personalizadas Amazon Rekognition.
- DeleteProjectVersion— Elimina un modelo de Amazon Rekognition Custom Labels.
- DescribeProjectVersions— Devuelve una lista de todos los modelos de Amazon Rekognition Custom Labels de un proyecto específico.

Etiquetas

- TagResource— Añade una o más etiquetas clave-valor a un modelo de etiquetas personalizadas de Amazon Rekognition.
- UntagResource— Elimina una o más etiquetas de un modelo de Amazon Rekognition Custom Labels.

Uso del modelo

- DetectCustomLabels— Analiza una imagen con su modelo de etiquetas personalizadas.
- StartProjectVersion— Inicia su modelo de etiquetas personalizadas.
- StopProjectVersion— Detiene su modelo de etiquetas personalizadas.

Historial de documentos de Etiquetas personalizadas de Amazon Rekognition

En la siguiente tabla se describen los cambios importantes en cada versión de la Guía para desarrolladores de Etiquetas personalizadas de Amazon Rekognition. Para recibir notificaciones sobre los cambios en esta documentación, puede suscribirse a una fuente RSS.

- Últimos cambios de la documentación: 19 de abril de 2023

Cambio	Descripción	Fecha
Se ha añadido el tema sobre la duración del modelo	Indica cómo obtener el número de horas de ejecución y las unidades de inferenci a utilizadas por un modelo. Para obtener más informaci ón, consulte Cómo obtener informes sobre la duración de la ejecución y las unidades de inferencia utilizadas.	19 de abril de 2023
Contenido de conjuntos de datos reorganizados	Se ha movido el contenido de creación del archivo de manifiesto al archivo de manifiesto. Se han movido los temas de conversión de conjuntos de datos a Convertir otros formatos de conjuntos de datos en un archivo de manifiesto.	20 de febrero de 2023
Se actualizó la guía de IAM para AWS WAF	Se ha actualizado la guía para implementar las prácticas recomendadas de IAM. Para obtener más información,	15 de febrero de 2023

	consulta prácticas recomendadas de seguridad en IAM.	
Ver la matriz de confusión de un modelo de clasificación	En la consola de Etiquetas personalizadas de Amazon Rekognition no aparece la matriz de confusión de un modelo de clasificación. En su lugar, puede usar el AWS SDK para obtener y mostrar una matriz de confusión. Para obtener más información, consulte Visualización de la matriz de confusión de un modelo.	4 de enero de 2023
Ejemplo de función de Lambda actualizada	El ejemplo de la función de Lambda ahora indica cómo analizar las imágenes transferidas desde un archivo local o un bucket de Amazon S3. Para obtener más información, consulte Análisis de imágenes con una función de AWS Lambda.	2 de diciembre de 2022
Etiquetas personalizadas de Amazon Rekognition ahora puede copiar modelos entrenados	Ahora puedes copiar un modelo entrenado de una AWS cuenta a otra AWS dentro de la misma AWS región. Para obtener más información, consulte Copia de un modelo de Etiquetas personalizadas de Amazon Rekognition (SDK).	16 de agosto de 2022

Etiquetas personalizadas de Amazon Rekognition ahora puede escalar automáticamente las unidades de inferencia.	Para hacer frente a los picos de demanda, Etiquetas personalizadas de Amazon Rekognition ahora puede escalar el número de unidades de inferencia que utiliza el modelo. Para obtener más información, consulte Ejecución de un modelo entrenado de Etiquetas personalizadas de Amazon Rekognition.	16 de agosto de 2022
Creación de un archivo de manifiesto a partir de un archivo CSV	Ahora puede hacer más sencilla la creación de un archivo de manifiesto mediante un script que lee la información de clasificación de imágenes a partir de un archivo CSV. Para obtener más información, consulte Creación de un archivo de manifiesto de un archivo CSV.	2 de febrero de 2022
Etiquetas personalizadas de Amazon Rekognition ahora administra conjuntos de datos con los proyectos	Puede usar los proyectos para administrar los conjuntos de datos de entrenamiento y de prueba que utilice para crear un modelo. Para obtener más información, consulte Cómo funciona Etiquetas personalizadas de Amazon Rekognition.	1 de noviembre de 2021

Amazon Rekognition Custom Labels está integrado con AWS CloudFormation	Se puede utilizar AWS CloudFormation para aprovisionar y configurar proyectos de etiquetas personalizadas de Amazon Rekognition. Para obtener más información, consulte Crear un proyecto con. AWS CloudFormation	21 de octubre de 2021
Entorno de guía de inicio actualizada	La consola de Etiquetas personalizadas de Amazon Rekognition ahora incluye tutoriales de vídeo y proyectos de ejemplo. Para obtener más información, consulte Introducción a Etiquetas personalizadas de Amazon Rekognition.	22 de julio de 2021
Información actualizada sobre los umbrales y el uso de métricas	Información sobre cómo elegir el valor de umbral deseado mediante el parámetro de entrada `MinConfidence` para `DetectCustomLabels` . Para obtener más información, consulte Análisis de una imagen con un modelo entrenado.	8 de junio de 2021
Se ha añadido AWS KMS key soporte	Ahora puede usar su propia clave de KMS para cifrar las imágenes de entrenamiento y de prueba. Para obtener más información, consulte Entrenamiento de un modelo.	19 de mayo de 2021

Se ha añadido la opción de etiquetado	Etiquetas personalizadas de Amazon Rekognition ahora admite el etiquetado. Puede usar etiquetas para identificar, organizar, buscar y filtrar sus modelos de Etiquetas personalizadas de Amazon Rekognition. Para obtener más información, consulte Etiquetado de un modelo.	25 de marzo de 2021
Tema de configuración actualizado	Información de configuración actualizada sobre cómo cifrar archivos de entrenamiento. Para obtener más información, consulte Paso 5: (Opcional) Cifrar archivos de entrenamiento.	18 de marzo de 2021
Se ha añadido un tema sobre la copia de conjuntos de datos	Información sobre cómo copiar un conjunto de datos a una AWS región diferente. Para obtener más información, consulte Copiar un conjunto de datos en una AWS región diferente.	5 de marzo de 2021

Se agregó el tema de transformación del manifiesto GroundTruth multietiqueta de Amazon SageMaker AI	Información sobre cómo transformar un manifiesto con formato de GroundTruth etiquetas múltiples de Amazon SageMaker AI en un archivo de manifiesto con formato de etiquetas personalizadas de Amazon Rekognition. Para obtener más información, consulta Transformar los archivos de manifiesto de SageMaker AI Ground Truth con varias etiquetas.	22 de febrero de 2021
Se ha añadido información sobre depuración de errores para el entrenamiento de modelos	Ahora puede usar los manifiestos de resultados de validación para obtener información detallada sobre la depuración de errores del entrenamiento de modelos. Para obtener más información, consulte Depuración de errores de un modelo de entrenamiento erróneo.	8 de octubre de 2020

Se ha añadido información y un ejemplo sobre la transformación COCO	Información sobre cómo transformar un conjunto de datos con formato de detección de objetos COCO en un archivo de manifiesto con formato de cuadro delimitador de Etiquetas personalizadas de Amazon Rekognition. Para obtener más información, consulte Transformación de conjuntos de datos COCO.	2 de septiembre de 2020
Etiquetas personalizadas de Amazon Rekognition ahora admite el entrenamiento con un solo objeto	Para crear un modelo de Etiquetas personalizadas de Amazon Rekognition que busque la ubicación de un único objeto, ahora puede crear un conjunto de datos que solo requiera una etiqueta. Para obtener más información, consulte Dibujo de cuadros delimitadores.	25 de junio de 2020
Se han añadido operaciones de borrado de proyectos y modelos	Ahora puede eliminar proyectos y modelos de Etiquetas personalizadas de Amazon Rekognition con la consola y con la API. Para obtener más información, consulte Eliminación de un modelo de Etiquetas personalizadas de Amazon Rekognition y Eliminación de un proyecto de Etiquetas personalizadas de Amazon Rekognition	1 de abril de 2020

Se han añadido ejemplos de código	Se han añadido ejemplos de Java para la creación de proyectos, el entrenamiento de modelos, la ejecución de modelos y el análisis de imágenes.	13 de diciembre de 2019
Nueva función y guía	Esta es la versión preliminar de la función de Etiquetas personalizadas de Amazon Rekognition y la Guía para desarrolladores de Etiquetas personalizadas de Amazon Rekognition.	3 de diciembre de 2019

Las traducciones son generadas a través de traducción automática. En caso de conflicto entre la traducción y la version original de inglés, prevalecerá la version en inglés.