aws

Guía para desarrolladores

Amazon Lookout for Vision

Amazon Lookout for Vision: Guía para desarrolladores

Table of Contents

Aviso de fin de soporte: el 31 de octubre de 2025, AWS dejaremos de ofrecer soporte a Amazon Lookout for Vision. Después del 31 de octubre de 2025, ya no podrás acceder a la consola Lookout for Vision ni a los recursos de Lookout for Vision. Para obtener más información, visita esta entrada de blog.

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.

¿Qué es Amazon Lookout for Vision?

Puede utilizar Amazon Lookout for Vision para encontrar defectos visuales en productos industriales de forma precisa y a escala. Utilice la visión artificial para identificar componentes faltantes en un producto industrial, daños en vehículos o estructuras, irregularidades en las líneas de producción e incluso defectos minúsculos en las obleas de silicio, o cualquier otro elemento físico en el que la calidad sea importante, como la falta de un condensador en las placas de circuito impreso.

Ventajas principales

Amazon Lookout for Vision ofrece los siguientes beneficios:

- Mejore los procesos de forma rápida y eficaz: puede utilizar Amazon Lookout for Vision para implementar la inspección basada en visión artificial en los procesos industriales de forma rápida y eficiente, a escala. Puede proporcionar tan solo 30 buenas imágenes de referencia y Lookout for Vision puede crear automáticamente un modelo ML personalizado para la detección de defectos. A continuación, puede procesar las imágenes de las cámaras IP, por lotes o en tiempo real, para identificar de forma rápida y precisa anomalías como abolladuras, grietas y arañazos.
- Aumente la calidad de la producción rápidamente: con Amazon Lookout for Vision, puede reducir los defectos en los procesos de producción en tiempo real. Identifica las anomalías visuales y las informa en un panel de control para que pueda tomar medidas rápidamente y evitar que se produzcan más defectos, lo que aumenta la calidad de la producción y reduce los costos.
- Reduzca los costes operativos: Amazon Lookout for Vision informa sobre las tendencias de sus datos de inspección visual, como la identificación de los procesos con la tasa de defectos más alta o la señalización de las variaciones recientes en los defectos. Con esta información, puede determinar si debe programar el mantenimiento de la línea de proceso o redirigir la producción a otra máquina antes de que se produzca un costoso e imprevisto tiempo de inactividad.

¿Es la primera vez que usa Amazon Lookout for Vision?

Si utiliza por primera vez Amazon Lookout for Vision, le recomendamos que lea las siguientes secciones en orden:

1. Configuración de Amazon Lookout for Vision: en esta sección, podrá ajustar los detalles de la cuenta.

2. Introducción a Amazon Lookout for Vision: en esta sección, aprenderá a crear su primer modelo de Amazon Lookout for Vision.

Configuración de Amazon Lookout for Vision

En esta sección, se registra una cuenta de AWS y se configura Amazon Lookout for Vision.

Para obtener información sobre las AWS regiones que admiten Amazon Lookout for Vision, consulte Amazon Lookout for Vision Endpoints and Quotas.

Temas

Paso 1: Cree una cuenta AWS

En este paso, se crea una AWS cuenta y se crea un usuario administrativo.

Temas

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 consiste en recibir una llamada telefónica o mensaje de texto e indicar 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 .

Paso 2: configuración de permisos

Para utilizar Amazon Lookout for Vision, necesita permisos de acceso a la consola de Lookout for Vision AWS , a las operaciones del SDK y al bucket de Amazon S3 que utiliza para el entrenamiento de modelos.

> **Note**
>
> Si solo usa las operaciones AWS del SDK, puede usar políticas que se ajusten a AWS las operaciones del SDK. Para obtener más información, consulte Configuración de permisos de SDK.

Temas

- Configurar el acceso a la consola con políticas AWS administradas
- Ajuste de permisos de bucket de Amazon S3

- Asignación de permisos

Configurar el acceso a la consola con políticas AWS administradas

Utilice las siguientes políticas AWS gestionadas para aplicar los permisos de acceso adecuados a las operaciones de la consola y el SDK de Amazon Lookout for Vision.

- AmazonLookoutVisionConsoleFullAccess— permite el acceso total a la consola de Amazon Lookout for Vision y a las operaciones del SDK. Necesita permisos `AmazonLookoutVisionConsoleFullAccess` para crear el bucket de consola. Para obtener más información, consulte Paso 3: Crear un bucket de consola.
- AmazonLookoutVisionConsoleReadOnlyAccess— permite el acceso de solo lectura a la consola Amazon Lookout for Vision y a las operaciones del SDK.

Para asignar permisos, consulte Asignación de permisos.

Para obtener información sobre las políticas AWS administradas, consulte Políticas administradas por AWS.

Ajuste de permisos de bucket de Amazon S3

Amazon Lookout for Vision utiliza un bucket de Amazon S3 para almacenar los siguientes archivos:

- Imágenes de conjuntos de datos: imágenes que se utilizan para entrenar un modelo. Para obtener más información, consulte Creación de un conjunto de datos.
- Archivos de manifiesto en formato Amazon SageMaker AI Ground Truth. Por ejemplo, el archivo de manifiesto generado por GroundTruth un trabajo de SageMaker IA. Para obtener más información, consulte Creación de un conjunto de datos mediante un archivo de manifiesto de Amazon SageMaker AI Ground Truth.
- El resultado del entrenamiento de modelos.

Si usa la consola, Lookout for Vision crea un bucket de Amazon S3 (bucket de consola) que puede utilizar para administrar sus proyectos. Las políticas administradas `LookoutVisionConsoleReadOnlyAccess` y `LookoutVisionConsoleFullAccess` incluyen permisos de acceso a Amazon S3 para el bucket de consola.

Puedes usar el depósito de la consola para almacenar imágenes de conjuntos de datos y archivos de manifiesto en formato SageMaker AI Ground Truth. Como alternativa, puede utilizar otro bucket de Amazon S3. El bucket debe ser propiedad de tu cuenta de AWS y debe estar ubicado en la AWS región en la que utilices Lookout for Vision.

Para utilizar otro bucket, agregue la siguiente política al usuario o grupo deseado. Sustituya `amzn-s3-demo-bucket` por el nombre del bucket deseado. Para obtener información sobre inclusión de las políticas de IAM, consulte Creación de políticas de IAM.

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

Para asignar permisos, consulte Asignación de permisos.

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

Para usar la consola de Amazon Lookout for Vision, necesita un bucket de Amazon S3 que se conoce como bucket de consola. El bucket de consola almacena lo siguiente:

- Imágenes que se cargan en un conjunto de datos con la consola.
- Resultados de entrenamiento para el entrenamiento de modelos que se inicia con la consola.
- Resultados de pruebas de detección.
- Archivos de manifiesto temporales que la consola crea cuando se usa para crear un conjunto de datos etiquetando automáticamente las imágenes en un bucket de S3. La consola no elimina los archivos de manifiesto.

Cuando abres la consola Amazon Lookout for Vision por primera vez en una región AWS nueva, Lookout for Vision crea el bucket de consola en tu nombre. Anote el nombre del bucket de la consola porque puede que necesite usarlo en las operaciones del AWS SDK o en las tareas de la consola, como la creación de un conjunto de datos.

Alternativamente, puede crear el bucket de consola usando Amazon S3. Utilice este enfoque si las políticas de bucket de Amazon S3 no permiten que la consola de Amazon Lookout for Vision cree correctamente el bucket de consola. Por ejemplo, una política que no permite la creación automática de un bucket de Amazon S3.

> **Note**
>
> Si solo utilizas el AWS SDK y no la consola Lookout for Vision, no necesitas crear el bucket de consola. Puede usar un bucket de S3 diferente con el nombre que elija.

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

Temas

- Creación del bucket de consola con la consola de Amazon Lookout for Vision
- Creación del bucket de consola con Amazon S3
- Configuración del bucket de consola

Creación del bucket de consola con la consola de Amazon Lookout for Vision

Utilice el siguiente procedimiento para crear el bucket de consola para una AWS región con la consola Amazon Lookout for Vision. Para obtener información sobre la configuración del bucket de S3 que habilitamos, consulte Configuración del bucket de consola.

Para crear el bucket de consola mediante la consola de Amazon Lookout for Vision

1. Asegúrese de que el usuario o grupo que está utilizando tenga permiso `AmazonLookoutVisionConsoleFullAccess`. Para obtener más información, consulte Paso 2: configuración de permisos.
2. Abre la consola https://console.aws.amazon.com/lookoutvision/Amazon Lookout for Vision en.
3. En la barra de navegación, elija seleccione una región. A continuación, elija la AWS región para la que desee crear el bucket de la consola.
4. Elija Comenzar.

5. Si es la primera vez que abre la consola en la región de AWS actual, realice lo siguiente en el cuadro de diálogo Primera configuración:

   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 Amazon Lookout for Vision cree el bucket de consola en su nombre.

   El cuadro de diálogo de configuración por primera vez no se muestra si el depósito de consola de la AWS región actual ya existe.

6. Cierre la ventana del navegador.

Creación del bucket de consola con Amazon S3

Puede utilizar la consola de Amazon S3 para crear el bucket de consola. Debe crear un bucket con el control de versiones Amazon S3 habilitado. Le recomendamos que utilice una configuración del ciclo de vida de Amazon S3 para eliminar las versiones no actuales (anteriores) de un objeto y eliminar las cargas multiparte incompletas. No recomendamos una configuración de ciclo de vida que elimine las versiones actuales de un objeto. Para obtener información sobre la configuración de los buckets de S3 que habilitamos para los buckets de consola que crea con la consola de Amazon Lookout for Vision, consulte Configuración del bucket de consola.

1. Decida la AWS región en la que desea crear un bucket de consola. Para obtener información sobre las regiones compatibles, consulte Puntos de conexión y cuotas de Amazon Lookout for Vision.
2. Cree un bucket siguiendo las instrucciones de la consola S3 que se encuentran en Crear un bucket.. Haga lo siguiente:

   a. En el paso 3, especifique un nombre de bucket que vaya precedido de `lookoutvision-`*`region`*`-`*`your-identifier`*. Cambie `region` al código de región que ha elegido en el paso anterior. Cambie `your-identifier` a un identificador único de su elección.

   b. En el paso 4, elige la AWS región que quieres usar.

3. Habilite el control de versiones para el bucket siguiendo las instrucciones de la consola de S3 que se encuentran en Habilitar el control de versiones en buckets.

4. (Opcional) Especifique una configuración de ciclo de vida para el bucket siguiendo las instrucciones de la consola de S3 que se encuentran en Configurar el ciclo de vida de un bucket. Haga lo siguiente para eliminar las versiones no actuales (anteriores) de un objeto y eliminar las cargas multiparte incompletas. No es necesario realizar los pasos 6, 8, 9 ni 10.

   a. Para el paso 5, seleccione Aplicar a todos los objetos del bucket.
   b. En el paso 7, seleccione Eliminar permanentemente las versiones no actuales de los objetos y Eliminar los objetos caducados, eliminar los marcadores o las cargas multiparte incompletas.
   c. En el paso 11, introduzca el número de días que deben transcurrir antes de eliminar las versiones no actuales de un objeto.
   d. En el paso 12, introduzca el número de días que deben transcurrir antes de eliminar las cargas multiparte incompletas.

Configuración del bucket de consola

Si crea el bucket de consola con la consola de Amazon Lookout for Vision, habilitamos los siguientes ajustes en el bucket de consola.

- Control de versiones de los objetos del bucket de consola.
- Cifrado del servidor de los objetos del bucket de consola.
- Una configuración de ciclo de vida para la eliminación de objetos no actuales (30 días) y cargas multiparte incompletas (3 días).
- Bloquear el acceso público al bucket de la consola.

Paso 4: Configura el AWS CLI y AWS SDKs

Los siguientes pasos le muestran cómo instalar AWS Command Line Interface (AWS CLI) y AWS SDKs. Los ejemplos de esta documentación utilizan Python y Java AWS SDKs. AWS CLI

Temas

- Instale los AWS SDK
- Concesión de acceso programático
- Configuración de permisos de SDK
- Llame a una operación de Amazon Lookout for Vision

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 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.
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 credenciales de usuario de IAM en la Guía del AWS Command Line Interface usuario. • Para obtener AWS SDKs información sobre las herramientas, consulte

¿Qué usuario necesita acceso programático?	Para	Mediante
		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 `lookoutvision-access`. Por ejemplo:

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

El usuario que representa el perfil debe tener permisos para llamar a las operaciones del SDK de Lookout for Vision y a AWS otras operaciones del SDK que se necesitan en los ejemplos. Para obtener más información, consulte Configuración de permisos de SDK. Para asignar permisos, consulte Asignación de permisos.

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 `lookoutvision-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.
- Dispositivo central Greengrass: usa certificados X.509 para conectarse a AWS IoT Core mediante protocolos de autenticación mutua de TLS. Estos certificados permiten que los dispositivos

interactúen con AWS IoT sin credenciales de AWS. El proveedor de credenciales AWS IoT autentifica a un intermediario mediante un certificado X.509 y emite credenciales de AWS con un token de seguridad temporal con privilegios limitados. 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 SDK de Amazon Lookout for Vision, necesita permisos de acceso a la API de Lookout for Vision y al bucket de Amazon S3 que utiliza para el entrenamiento de modelos.

Temas

- Otorgar permisos de operaciones del SDK
- Concesión de permisos de bucket de Amazon S3
- 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 DetectAnomalies, necesitas permiso para actuar`lookoutvision:DetectAnomalies`. 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.

- AmazonLookoutVisionFullAccess— permite el acceso total a las operaciones del SDK de Amazon Lookout for Vision.
- AmazonLookoutVisionReadOnlyAccess— permite el acceso a las operaciones del SDK de solo lectura.

Las políticas administradas de la consola también proporcionan permisos de acceso para las operaciones de SDK. Para obtener más información, consulte Paso 2: configuración de permisos.

Para obtener información sobre las políticas AWS administradas, consulte Políticas administradas por AWS.

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.

> **Note**
>
> Las instrucciones de introducción requieren permisos `s3:PutObject`. Para obtener más información, consulte Paso 1: Crear el archivo de manifiesto y cargar las imágenes.

Para asignar permisos, consulte Asignación de permisos.

Concesión de permisos de bucket de Amazon S3

Para entrenar un modelo, necesita un bucket de Amazon S3 con los permisos adecuados para almacenar las imágenes, los archivos de manifiesto y el resultado del entrenamiento. El bucket debe ser propiedad de su cuenta de AWS y debe estar ubicado en la región de AWS en la que utilice Amazon Lookout for Vision.

Las políticas administradas únicamente por el SDK (`AmazonLookoutVisionFullAccessyAmazonLookoutVisionReadOnlyAccess`) no incluyen los permisos de los buckets de Amazon S3, por lo que debe aplicar la siguiente política de permisos para acceder a los buckets que utiliza, incluidos los buckets de consola existentes.

Las políticas administradas de la consola (`AmazonLookoutVisionConsoleFullAccess` y `AmazonLookoutVisionConsoleReadOnlyAccess`) incluyen permisos de acceso al bucket de la consola. Si accede al bucket de consola con operaciones de SDK y tiene permisos de políticas administradas de la consola, no necesita usar la siguiente política. Para obtener más información, consulte Paso 2: configuración de permisos.

Determinación de los permisos de las tareas

Utilice la siguiente información para decidir qué permisos son necesarios para las tareas que desee realizar.

Creación de un conjunto de datos

Para crear un conjunto de datos con CreateDataset, necesita los siguientes permisos.

- `s3:GetBucketLocation`: permite a Lookout for Vision validar que su bucket se encuentra en la misma región en la que utiliza Lookout for Vision.
- `s3:GetObject`: permite el acceso al archivo de manifiesto especificado en el parámetro de entrada `DatasetSource`. Si quiere especificar una versión exacta de un objeto S3 del archivo de manifiesto, también necesita especificarla `s3:GetObjectVersion` en el archivo de manifiesto. Para obtener más información, consulte Usar el control de versiones en buckets de S3.

Creación de un modelo

Para crear un modelo con CreateModel, necesita los siguientes permisos.

- `s3:GetBucketLocation`: permite a Lookout for Vision validar que su bucket se encuentra en la misma región en la que utiliza Lookout for Vision.
- `s3:GetObject`: permite el acceso a las imágenes especificadas en los conjuntos de datos de entrenamiento y prueba del proyecto.
- `s3:PutObject`: permite almacenar los resultados del entrenamiento en el bucket especificado. La ubicación del bucket de salida se especifica en el `OutputConfig` parámetro. Si lo desea, puede aplicar los permisos a solo las claves de objeto especificadas en el campo `Prefix` del campo de entrada `S3Location`. Para obtener más información, consulte OutputConfig.

Acceso a imágenes, archivos de manifiesto y resultados de entrenamiento

No se requieren permisos de bucket de Amazon S3 para ver las respuestas de las operaciones de Amazon Lookout for Vision. Sin embargo, sí necesita permiso `s3:GetObject` si quiere acceder a las imágenes, los archivos de manifiesto y los resultados de entrenamiento a los que se hace referencia en las respuestas de las operaciones. Si está accediendo a un objeto de Amazon S3 versionado, necesita `s3:GetObjectVersion` permiso.

Configuración de la política del bucket de Amazon S3

Puede usar la siguiente política para especificar los permisos del bucket de Amazon S3 necesarios para crear un conjunto de datos (`CreateDataset`), crear un modelo (`CreateModel`) y acceder a imágenes, archivos de manifiesto y resultados de entrenamiento. Cambie el valor *`amzn-s3-demo-bucket`* de por el nombre del depósito que desee utilizar.

Puede ajustar la política a sus necesidades. Para obtener más información, consulte Determinación de los permisos de las tareas. Añada la política al usuario deseado. Para obtener más información, consulte Creación de políticas de IAM.

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

Para asignar permisos, consulte Asignación de permisos.

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.

Llame a una operación de Amazon Lookout for Vision

Ejecute el siguiente código para confirmar que puede realizar llamadas a la API de Amazon Lookout for Vision. 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 `ListProjects`.

En general, para llamar a una función de ejemplo es necesario un cliente de Lookout for Vision del AWS SDK y cualquier otro parámetro obligatorio. El cliente del AWS SDK Lookout for Vision 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 Amazon Lookout for Vision no esté disponible en AWS todas las regiones.

Para llamar a una operación de Amazon Lookout for Vision

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

   CLI

   Use el comando `list-projects` para ver los proyectos en su cuenta.

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

Python

Este código se ha tomado del GitHub repositorio de ejemplos del SDK de AWS documentación. Consulte el ejemplo completo aquí.

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


from botocore.exceptions import ClientError
import boto3

class GettingStarted:

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

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

    GettingStarted.list_projects(lookoutvision_client)
```

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

Java V2

Este código se ha tomado del GitHub repositorio de ejemplos del SDK de AWS documentación. Consulte el ejemplo completo aquí.

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

package com.example.lookoutvision;

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

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

public class GettingStarted {

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

    /**
     * Lists the Amazon Lookoutfor Vision projects in the current AWS account
 and
     * AWS Region.
     *
     * @param lfvClient   An Amazon Lookout for Vision client.
     * @return List<ProjectMetadata> Metadata for each project.
     */
```

```
   public static List<ProjectMetadata> listProjects(LookoutVisionClient
lfvClient)
           throws LookoutVisionException {

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

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

       ListProjectsIterable projects =
lfvClient.listProjectsPaginator(listProjectsRequest);

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

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

       return projectMetadata;

   }

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

       try {

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

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

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

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

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

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

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

    }

}
```

Paso 5: (opcional) Usar su propia clave de AWS Key Management Service

Puede utilizar AWS Key Management Service (KMS) para administrar las claves de las imágenes de entrada y los vídeos que almacena en los buckets de Amazon S3.

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 clave de AWS Key Management Service (KMS). Para obtener más información, consulte Conceptos de AWS Key Management Service.

Si quiere usar su propia clave de KMS, utilice la siguiente política para especificarla. Cambie *kms_key_arn* al ARN de la clave de KMS (o ARN del alias de KMS) que desee usar. Como alternativa, especifique * para usar cualquier clave de KMS. Para obtener más información sobre asignar la política a un usuario o rol, consulte Crear políticas de IAM.

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

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

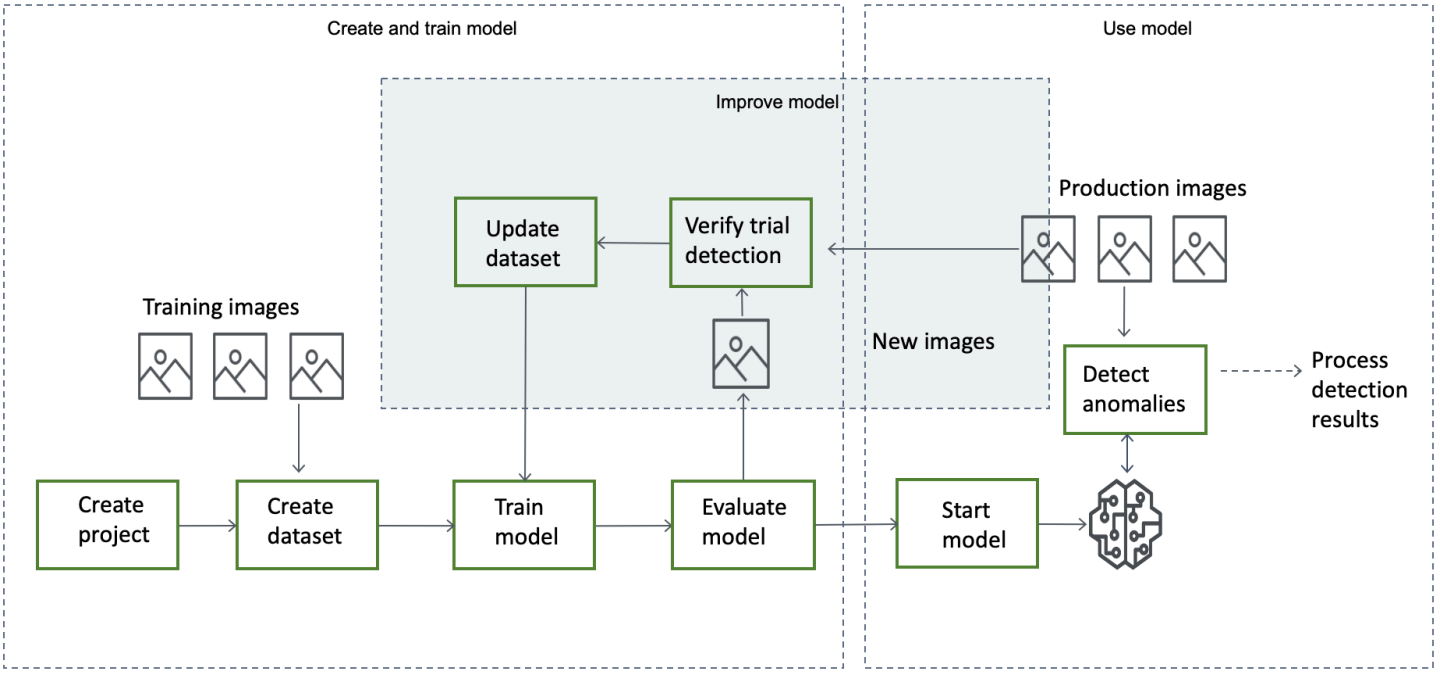
Análisis Amazon Lookout for Vision

Puede utilizar Amazon Lookout for Vision para encontrar defectos visuales en productos industriales de forma precisa y a escala, para tareas como:

- Detección de piezas dañadas: detecte los daños en la calidad de la superficie, el color y la forma de un producto durante el proceso de fabricación y ensamblaje.
- Identificación de los componentes faltantes: determine los componentes faltantes en función de la ausencia, presencia o ubicación de los objetos. Por ejemplo, falta un condensador en una placa de circuito impreso.
- Detectar problemas del proceso: detecte defectos con patrones repetitivos, como arañazos repetidos en el mismo punto de una oblea de silicio.

Con Lookout for Vision se crea un modelo de visión artificial que predice la presencia de anomalías en una imagen. El usuario proporciona las imágenes que Amazon Lookout for Vision utiliza para entrenar y probar su modelo. Amazon Lookout for Vision proporciona métricas que puede utilizar para evaluar y mejorar su modelo entrenado. Puede alojar el modelo entrenado en la AWS nube o puede implementar el modelo en un dispositivo perimetral. Una simple operación de API devuelve las predicciones que realiza el modelo.

El flujo de trabajo general para crear, evaluar y usar un modelo es el siguiente:



Temas

Elija su tipo de modelo

Antes de poder crear un modelo, debe decidir qué tipo de modelo desea. Puede crear dos tipos de modelo: clasificación de imágenes y segmentación de imágenes. El usuario decide el tipo de modelo que desea crear en función del caso de uso.

Ajuste de un modelo de clasificación de imágenes

Si solo necesita saber si una imagen contiene una anomalía, pero no necesita saber su ubicación, cree un modelo de clasificación de imágenes. Un modelo de clasificación de imágenes predice si una imagen contiene una anomalía. La predicción incluye la confianza del modelo en la precisión de la predicción. El modelo no proporciona ninguna información sobre la ubicación de las anomalías encontradas en la imagen.

Métricas del modelo de segmentación de imágenes

Si necesita conocer la ubicación de una anomalía, como la ubicación de un arañazo, cree un modelo de segmentación de imágenes. Los modelos de Amazon Lookout for Vision utilizan la segmentación semántica para identificar los píxeles de una imagen en los que se encuentran los tipos de anomalías (como un arañazo o una pieza faltante).

> **Note**
>
> Un modelo de segmentación semántica localiza diferentes tipos de anomalías. No proporciona información de instancia para anomalías individuales. Por ejemplo, si una imagen contiene dos abolladuras, Lookout for Vision devuelve información sobre ambas abolladuras en una sola entidad que representa el tipo de anomalía de abolladura.

Un modelo de segmentación de Amazon Lookout for Vision predice lo siguiente:

Clasificación

El modelo devuelve una clasificación para una imagen analizada (normal/anomalía), que incluye la confianza del modelo en la predicción. La información de clasificación se calcula por separado de la información de segmentación y no se debe suponer que existe una relación entre ambas.

Segmentación

El modelo devuelve una máscara de imagen que marca los píxeles en los que se producen las anomalías en la imagen. Los diferentes tipos de anomalías se codifican por colores según el color asignado a la etiqueta de anomalía en el conjunto de datos. Una etiqueta de anomalía representa el tipo de anomalía. Por ejemplo, la máscara azul de la siguiente imagen marca la ubicación de un tipo de anomalía por arañazo que se encuentra en un automóvil.

El modelo devuelve el código de color de cada etiqueta de anomalía de la máscara. El modelo también devuelve el porcentaje de cobertura de la imagen que tiene una etiqueta de anomalía.

Con un modelo de segmentación de Lookout for Vision, puede utilizar varios criterios para analizar los resultados del análisis del modelo. Por ejemplo:

- Ubicación de las anomalías: si necesita saber la ubicación de las anomalías, utilice la información de segmentación para ver máscaras que las cubran.
- Tipos de anomalías: utilice la información de segmentación para decidir si una imagen contiene más de un número aceptable de tipos de anomalías.
- Área de cobertura: utilice la información de segmentación para decidir si un tipo de anomalía cubre más de un área aceptable de una imagen.
- Clasificación de imágenes: si no necesita saber la ubicación de las anomalías, utilice la información de clasificación para determinar si una imagen contiene anomalías.

Para ver el código de ejemplo, consulte Detección de anomalías en una imagen.

Una vez que haya decidido qué tipo de modelo desea, cree un proyecto y un conjunto de datos para administrar el modelo. Con las etiquetas, puede clasificar las imágenes como normales o anómalas. Las etiquetas también identifican la información de segmentación, como las máscaras y los tipos de anomalías. La forma en que se etiquetan las imágenes en el conjunto de datos determina el tipo de modelo que Lookout for Vision crea.

Etiquetar un modelo de segmentación de imágenes es más complejo que etiquetar un modelo de clasificación de imágenes. Para entrenar un modelo de segmentación, hay que clasificar las imágenes de entrenamiento como normales o anómalas. También debe definir máscaras de anomalías y tipos de anomalías para cada imagen anómala. Un modelo de clasificación solo requiere que identifique las imágenes de entrenamiento como normales o anómalas.

Cree el modelo

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

Crear un proyecto

Cree un proyecto para administrar los conjuntos de datos y los modelos que cree. Un proyecto debe utilizarse para un único caso de uso, como la detección de anomalías en un único tipo de pieza de la máquina.

Use el panel de control para obtener una visión general de los proyectos. Para obtener más información, consulte Uso del panel de Amazon Lookout for Vision.

Más información: Cree su proyecto.

Creación de un conjunto de datos

Para entrenar un modelo, Amazon Lookout for Vision necesita imágenes de objetos normales y anómalos para su caso de uso. Usted suministra estas imágenes en un conjunto de datos.

Un conjunto de datos es un conjunto de imágenes y etiquetas que describen esas imágenes. Las imágenes deben representar un único tipo de objeto en el que puedan producirse anomalías. Para obtener más información, consulte Preparación de imágenes para un conjunto de datos.

Con Amazon Lookout for Vision, puede tener un proyecto que utilice un único conjunto de datos o un proyecto que tenga conjuntos de datos de entrenamiento y prueba independientes. Recomendamos

usar un proyecto de conjunto de datos único, a menos que desee tener un mayor control sobre el entrenamiento, las pruebas y el ajuste del rendimiento.

Para crear un conjunto de datos, importa las imágenes. Según cómo importe las imágenes, es posible que las imágenes también estén etiquetadas. De lo contrario, utilice la consola para etiquetar las imágenes.

Importación de imágenes

Si crea el conjunto de datos con la consola de Lookout for Vision, puede importar las imágenes de una de las siguientes maneras:

- Importar imágenes de un equipo local. Las imágenes no están etiquetadas.
- Importar imágenes de un bucket de S3. Amazon Lookout for Vision puede clasificar las imágenes mediante los nombres de las carpetas que las contienen. Use `normal` para imágenes normales. Se utiliza `anomaly` para imágenes anómalas. No puede asignar etiquetas de segmentación automáticamente.
- Importa un archivo de manifiesto de Amazon SageMaker AI Ground Truth. Las imágenes de un archivo de manifiesto están etiquetadas. Puede crear e importar su propio archivo de manifiesto. Si tiene muchas imágenes, considere utilizar el servicio de etiquetado SageMaker AI Ground Truth. A continuación, importas el archivo de manifiesto de salida del trabajo de Amazon SageMaker AI Ground Truth.

Etiquetado de imágenes

Las etiquetas describen una imagen de un conjunto de datos. Las etiquetas especifican si una imagen es normal o anómala (clasificación). Las etiquetas también describen la ubicación de las anomalías en una imagen (segmentación).

Si las imágenes no están etiquetadas, puede usar la consola para etiquetarlas.

Las etiquetas que asigna a las imágenes en el conjunto de datos determina el tipo de modelo que Lookout for Vision crea:

Clasificación de imágenes

Para crear un modelo de clasificación de imágenes, utilice la consola de Lookout for Vision para clasificar las imágenes del conjunto de datos como normales o anómalas.

También puede usar la operación `CreateDataset` para crear un conjunto de datos a partir de un archivo de manifiesto que incluya información de clasificación.

Segmentación de imágenes

Para crear un modelo de clasificación de imágenes, utilice la consola de Lookout for Vision para clasificar las imágenes del conjunto de datos como normales o anómalas. También especificas máscaras de píxeles para las áreas anómalas de la imagen (si existen), así como una etiqueta de anomalía para las máscaras de anomalías individuales.

También puede usar la operación `CreateDataset` para crear un conjunto de datos a partir de un archivo de manifiesto que incluya información de segmentación y clasificación.

Si su proyecto tiene conjuntos de datos de entrenamiento y de prueba independientes, Lookout for Vision usa el conjunto de datos de entrenamiento para aprender y determinar el tipo de modelo. Debería etiquetar las imágenes del conjunto de datos de prueba de la misma manera.

Más información: Cómo crear un conjunto de datos.

Entrenamiento de su modelo

El entrenamiento crea un modelo y lo entrena para predecir la presencia de anomalías en las imágenes. Se creará una nueva versión del modelo cada vez que se entrena.

Al inicio del entrenamiento, Amazon Lookout for Vision elija el algoritmo más adecuado para entrenar el modelo. El modelo se entrena y, a continuación, se prueba. Si entrenas un único proyecto de conjunto de datos, el conjunto de datos se divide internamente para crear un conjunto de datos de entrenamiento y un conjunto de datos de prueba. Introducción a Amazon Lookout for Vision También puede crear un proyecto que tenga conjuntos de datos de entrenamiento y de prueba independientes. En esta configuración, Amazon Lookout for Vision entrena el modelo con el conjunto de datos de entrenamiento y prueba el modelo con el conjunto de datos de prueba.

> **Important**
>
> Se le cobrará por el tiempo que se tarde en entrenar correctamente un modelo.

Más información: Entrene su modelo.

Evaluar el modelo

Evalúe el rendimiento del modelo mediante las métricas de rendimiento creadas durante las pruebas.

Al utilizar las métricas de rendimiento, puede comprender mejor el rendimiento de su modelo entrenado y decidir si está preparado para usarlo en producción.

Más información: Cómo mejorar su modelo.

Si las métricas de rendimiento indican que es necesario realizar mejoras, puede añadir más datos de entrenamiento realizando una tarea de pruebas de detección con imágenes nuevas. Una vez finalizada la tarea, puede verificar los resultados y añadir las imágenes verificadas a su conjunto de datos de entrenamiento. Como alternativa, puede añadir nuevas imágenes de entrenamiento directamente al conjunto de datos. A continuación, vuelve a entrenar el modelo y vuelve a comprobar las métricas de rendimiento.

Más información: Verificar el modelo con una tarea de detección de prueba.

Use su modelo

Antes de poder usar el modelo en la AWS nube, hay que iniciarlo con la StartModeloperación. Puede obtener el comando `StartModel` CLI para su modelo desde la consola.

Más información: Inicie el modelo.

Un modelo entrenado de Amazon Lookout for Vision predice si una imagen de entrada contiene contenido normal o anómalo. Si su modelo es un modelo de segmentación, la predicción incluye una máscara de anomalías que marca los píxeles en los que se encuentran las anomalías.

Para hacer una predicción con su modelo, llame a la DetectAnomaliesoperación y pase una imagen de entrada desde su ordenador local. Puede obtener el comando CLI que llama `DetectAnomalies` desde la consola.

Más información: Detecta anomalías en una imagen.

> **Important**
>
> Se le cobrará por el tiempo de ejecución del modelo.

Si ya no utiliza el modelo, utilice la StopModeloperación para detenerlo. Puede obtener el comando CLI de la consola.

Más información: Detenga su modelo.

Use su modelo en un dispositivo de borde

Puedes usar un modelo Lookout for Vision en AWS IoT Greengrass Version 2 un dispositivo principal.

Más información: Uso del modelo Amazon Lookout for Vision en un dispositivo de borde.

Cómo usar el panel

Puede utilizar el panel de control para obtener una visión general de todos sus proyectos e información general de los proyectos individuales.

Más información: Usar el panel de control.

Introducción a Amazon Lookout for Vision

Antes de empezar con la Introducción, le recomendamos que lea Análisis Amazon Lookout for Vision.

Las instrucciones de introducción muestran cómo utilizar la creación de un modelo de segmentación de imágenes de ejemplo. Si desea crear un ejemplo de modelo de clasificación de imágenes, consulte Conjunto de datos de clasificación de imágenes.

Si quiere probar rápidamente un modelo de ejemplo, le ofrecemos ejemplos de imágenes de entrenamiento e imágenes de máscaras. También proporcionamos un script de Python que crea un archivo de manifiesto de segmentación de imágenes. Utilice el archivo de manifiesto para crear un conjunto de datos para su proyecto y no necesita etiquetar las imágenes en el conjunto de datos. Al crear un modelo con sus propias imágenes, debe etiquetar las imágenes en el conjunto de datos. Para obtener más información, consulte Creación de un conjunto de datos.

Las imágenes que proporcionamos son de cookies normales y anómalas. Una cookie anómala tiene una grieta en la forma de la cookie. El modelo que entrenes con las imágenes predice una clasificación (normal o anómala) y encuentra el área (máscara) de las grietas en una cookie anómala, como se muestra en el siguiente ejemplo.

Temas

Paso 1: Crear el archivo de manifiesto y cargar las imágenes

En este procedimiento, clona el repositorio de documentación de Amazon Lookout for Vision en su equipo. A continuación, utilice un script de Python (versión 3.7 o superior) para crear un archivo de manifiesto y cargar las imágenes de entrenamiento y las imágenes de la máscara en la ubicación de Amazon S3 que especifique. Use el archivo de manifiesto para crear el modelo. Más adelante, utilice imágenes de prueba en el repositorio local para probar el modelo.

Crear el archivo de manifiesto y cargar las imágenes

1. Configure Amazon Lookout for Vision siguiendo las instrucciones de Configuración de Amazon Lookout for Vision. Asegúrese de instalar el AWS SDK for Python.
2. En la AWS región en la que quieras usar Lookout for Vision, crea un bucket de S3.
3. En el bucket de Amazon S3, cree una carpeta llamada `getting-started`.
4. Anote la URI de Amazon S3 y el nombre de recurso de Amazon (ARN) de la carpeta. Los usa para configurar los permisos y ejecutar el script.
5. Asegúrese de que el usuario que ejecuta el script tiene permisos para llamar a la operación `s3:PutObject`. Puede utilizar la siguiente política. Para asignar permisos, consulte Asignación de permisos.

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

6. Asegúrese de tener un nombre de perfil local `lookoutvision-access` y de que el usuario del perfil tenga el permiso del paso anterior. Para obtener más información, consulte Uso de un perfil en su equipo local.
7. Descargue el archivo zip, getting-started.zip. El archivo zip contiene el conjunto de datos de introducción y el script de configuración.

8. Descomprima el archivo `getting-started.zip`.
9. En el símbolo del sistema, ejecute lo siguiente:

   a. Vaya a la carpeta `getting-started`.

   b. Ejecute el siguiente comando para crear un archivo de manifiesto y cargar las imágenes de entrenamiento y las máscaras de imagen en la ruta de Amazon S3 que indicó en el paso 4.

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

   c. Cuando se complete el script, anote la ruta al archivo `train.manifest` que aparece después de `Create dataset using manifest file:`. La ruta debería ser similar a `s3://path to getting started folder/manifests/train.manifest`.

Paso 2: crear el modelo

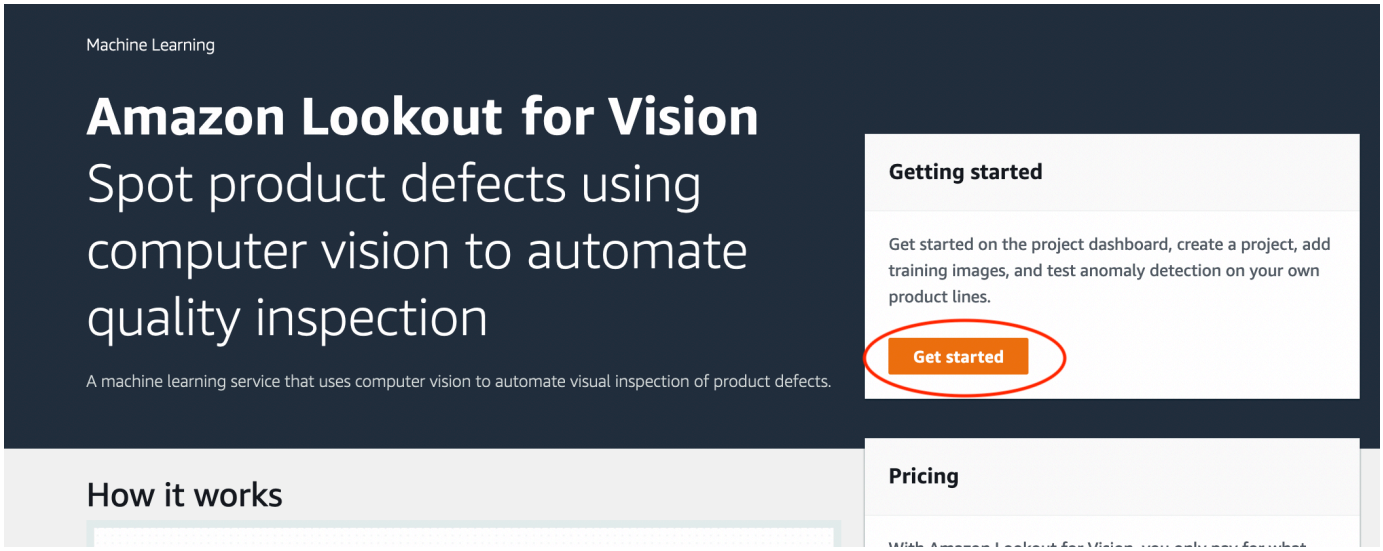
En este procedimiento, crea un proyecto y un conjunto de datos con las imágenes y el archivo de manifiesto que cargó anteriormente en su bucket de Amazon S3. A continuación, debe crear el modelo y ver los resultados de la evaluación del entrenamiento del modelo.

Como el conjunto de datos se crea a partir del archivo de manifiesto de introducción, no es necesario etiquetar las imágenes del conjunto de datos. Cuando crea un conjunto de datos con sus propias imágenes, sí que necesita etiquetar las imágenes. Para obtener más información, consulte Etiquetado de imágenes.

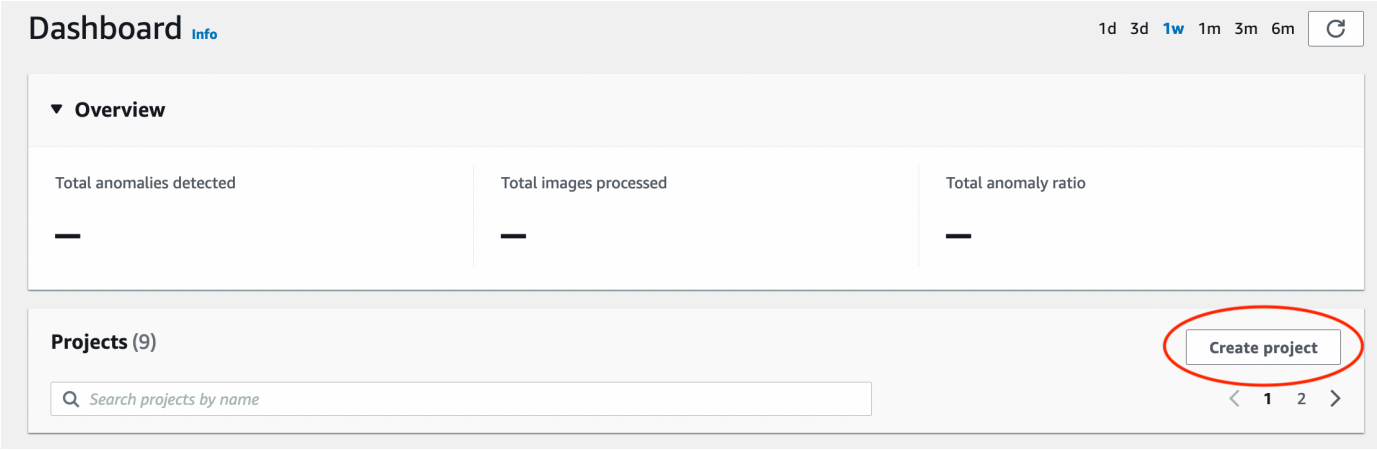
> **Important**
>
> Solo se le cobrará cuando termine de entrenar por completo un modelo.

Para crear un modelo

1. Abre la consola https://console.aws.amazon.com/lookoutvision/Amazon Lookout for Vision en.
2. Asegúrese de estar en la misma AWS región en la que creó el bucket de Amazon S3Paso 1: Crear el archivo de manifiesto y cargar las imágenes. Para cambiar de región de , en la barra de navegación, elija el nombre de la región mostrada actualmente. A continuación, elija la región a la que desea cambiar.
3. Elija Comenzar.



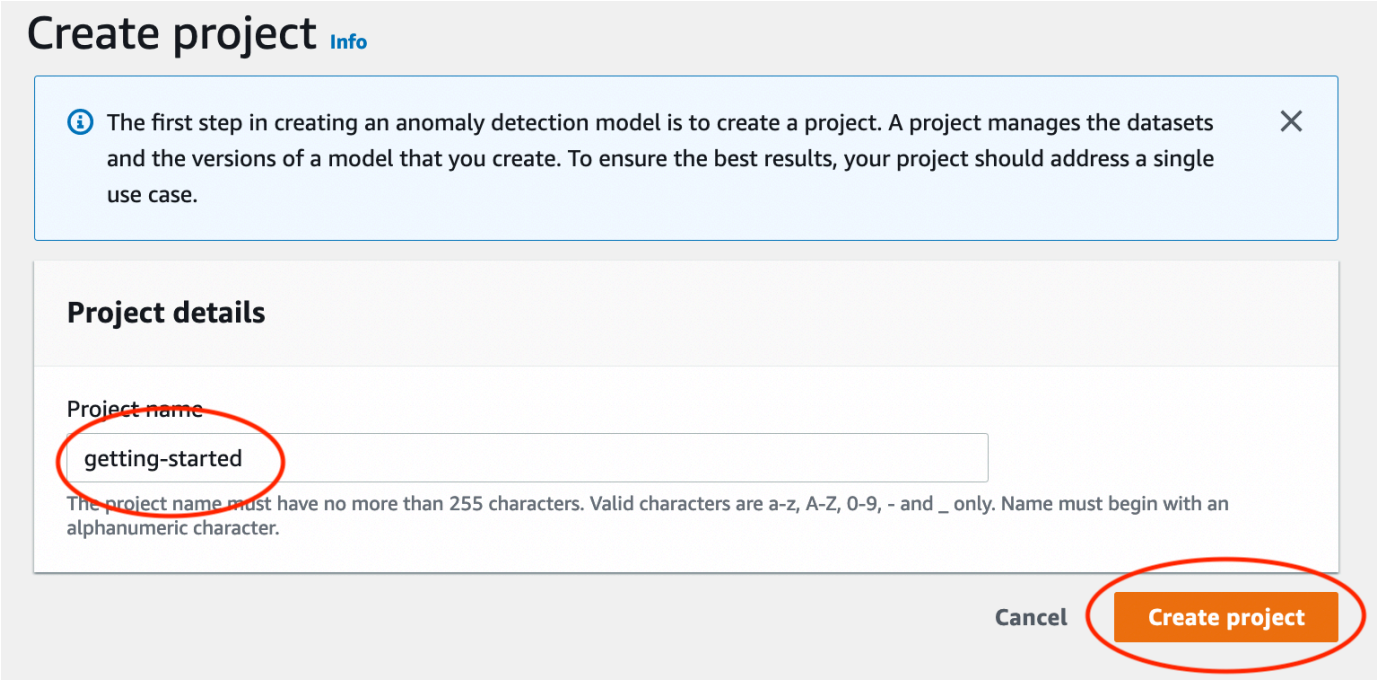
4. En la sección Proyectos, elija Crear proyecto.



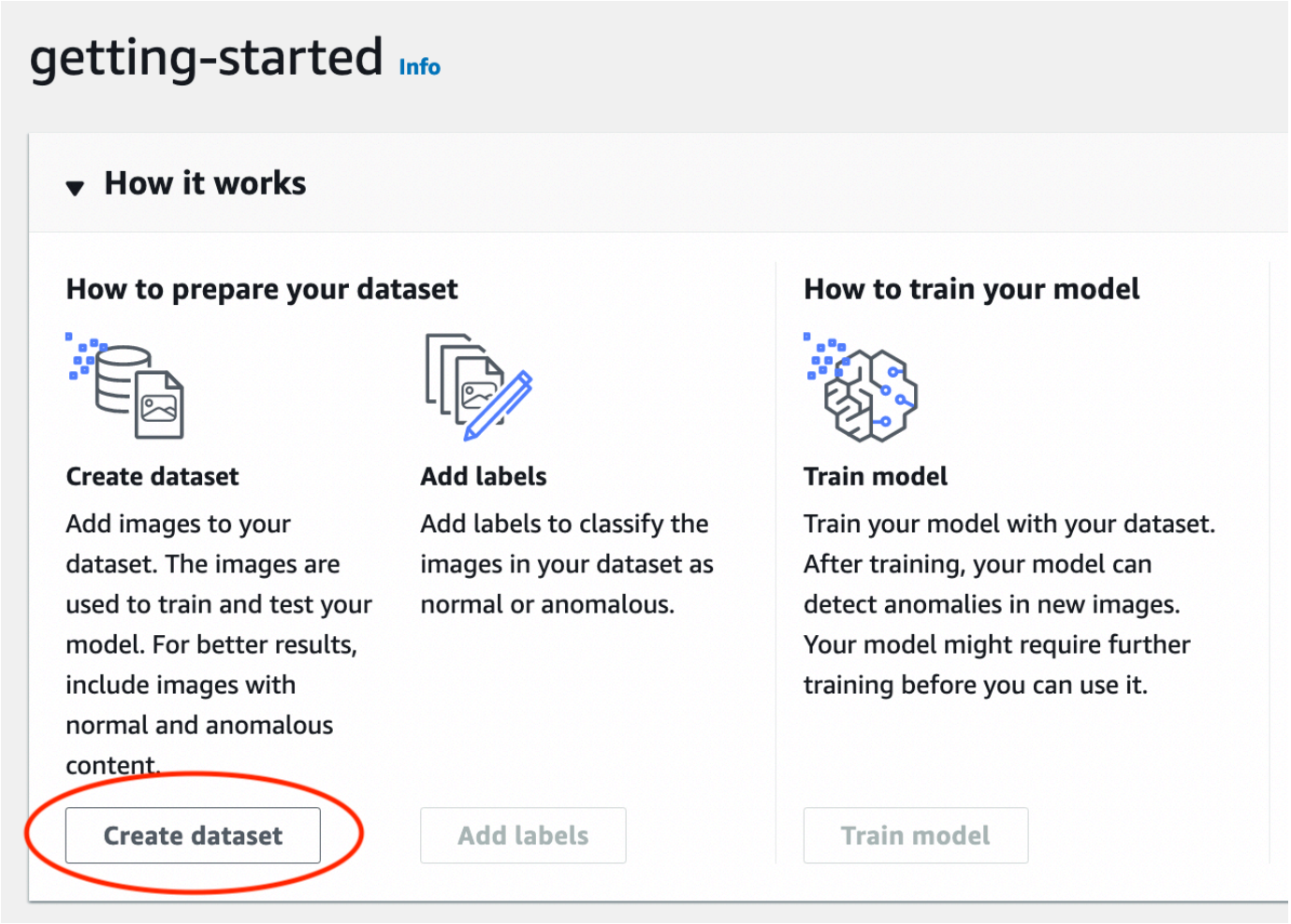
5. En la página Crear proyecto, haga lo siguiente:

   a. En Nombre del proyecto, introduzca `getting-started`.

   b. Elija Crear proyecto.



6. En la página del proyecto, en la sección Cómo funciona, seleccione Crear conjunto de datos.



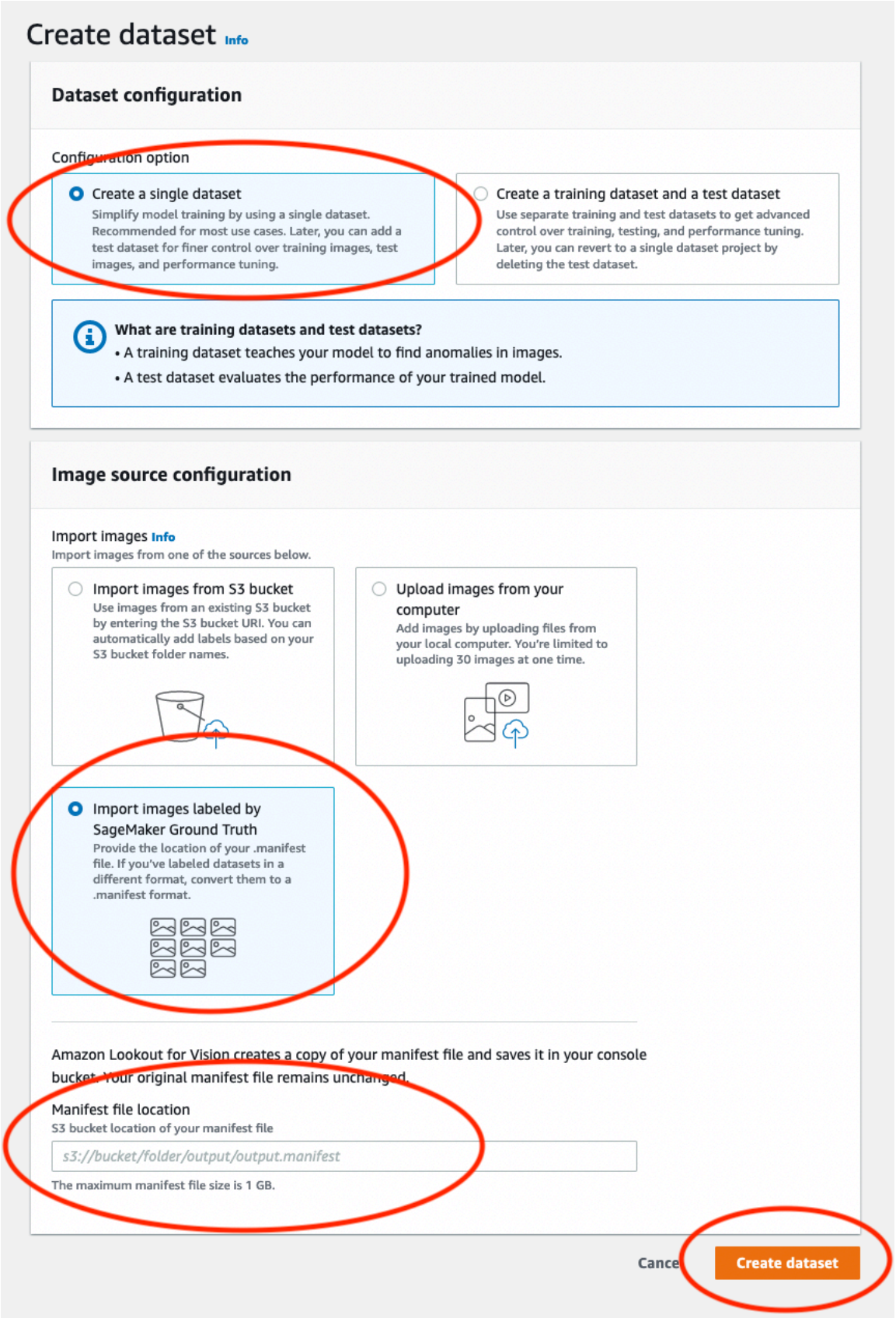
7. En la página Crear conjunto de datos, haga lo siguiente:

   a. Elija Crear un solo conjunto de datos.

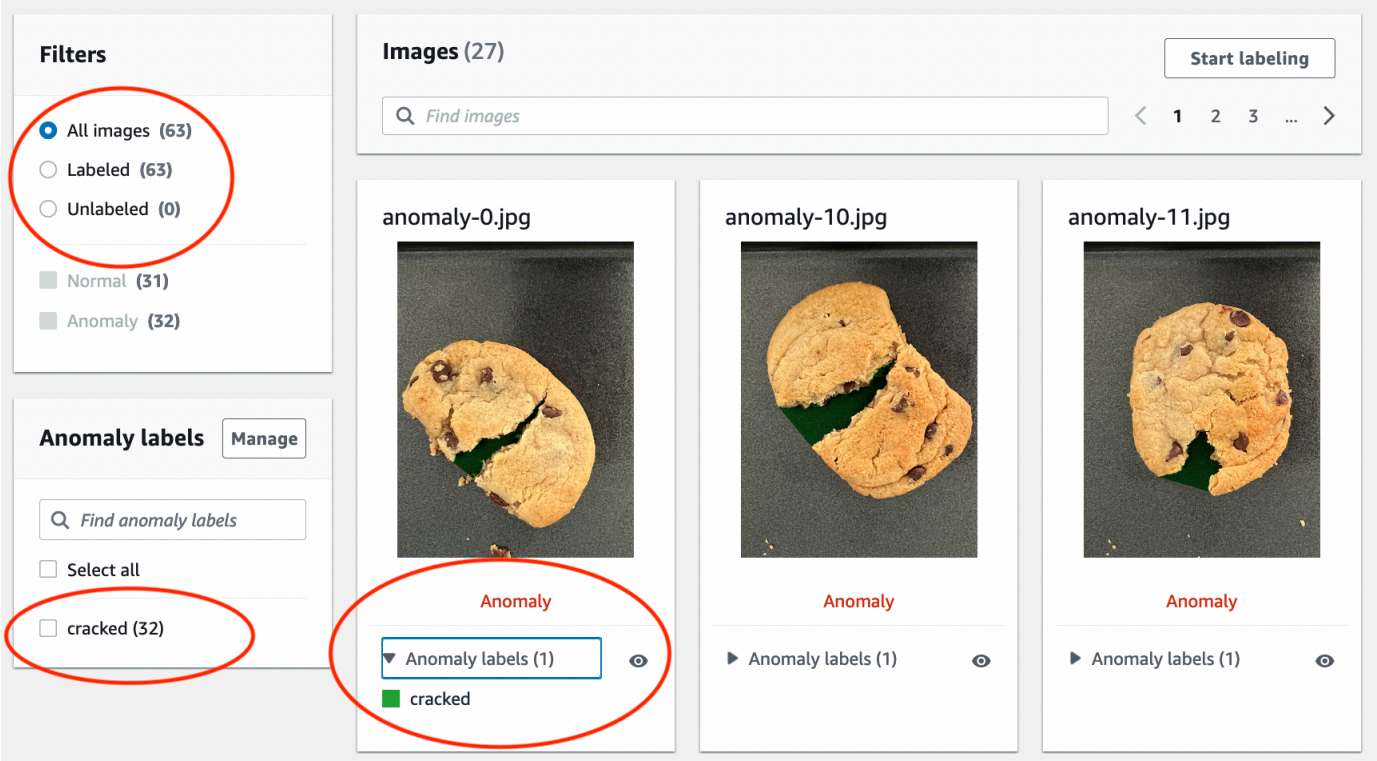
   b. En la sección Configuración de la fuente de la imagen, selecciona Importar imágenes etiquetadas por SageMaker Ground Truth.

   c. Para la ubicación del archivo .manifest, introduzca la ubicación de Amazon S3 que indicó en el paso 6.c de Paso 1: Crear el archivo de manifiesto y cargar las imágenes. La ubicación de Amazon S3 debe ser similar a `s3://path to getting started folder/manifests/train.manifest`

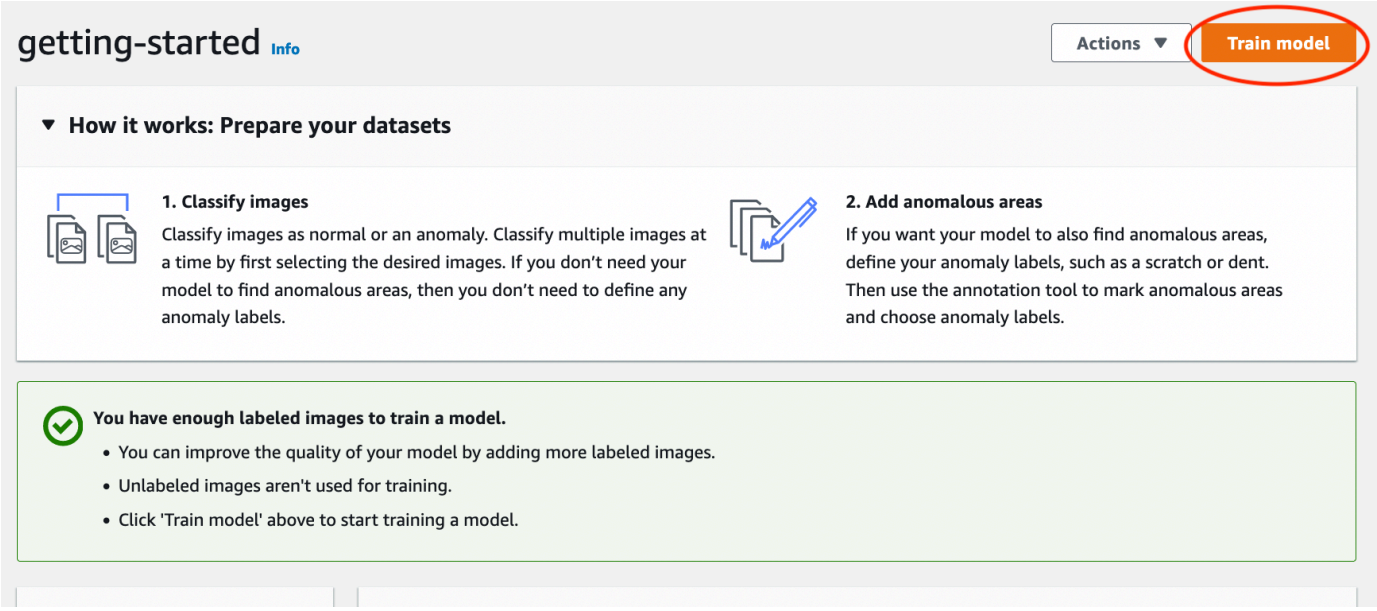
   d. Elija Crear conjunto de datos.


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

8. En la página de detalles del proyecto, en la sección Imágenes, consulte las imágenes del conjunto de datos. Puede ver la información de clasificación y segmentación de imágenes (etiquetas de máscara y anomalía) de cada imagen del conjunto de datos. También puede buscar imágenes, filtrarlas por estado de etiquetado (etiquetadas/sin etiquetar) o filtrar imágenes por las etiquetas de anomalías que se les hayan asignado.

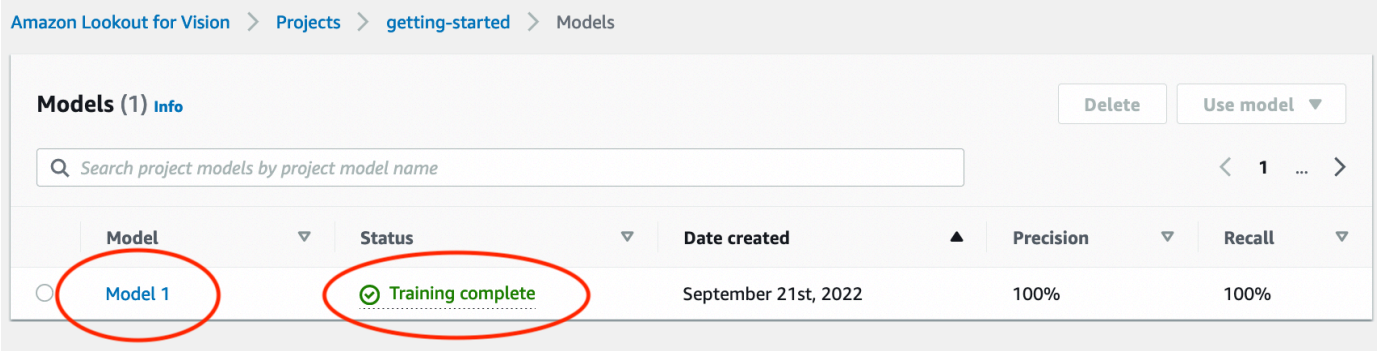


9. En la página de detalles del proyecto, elija Entrenar modelo.



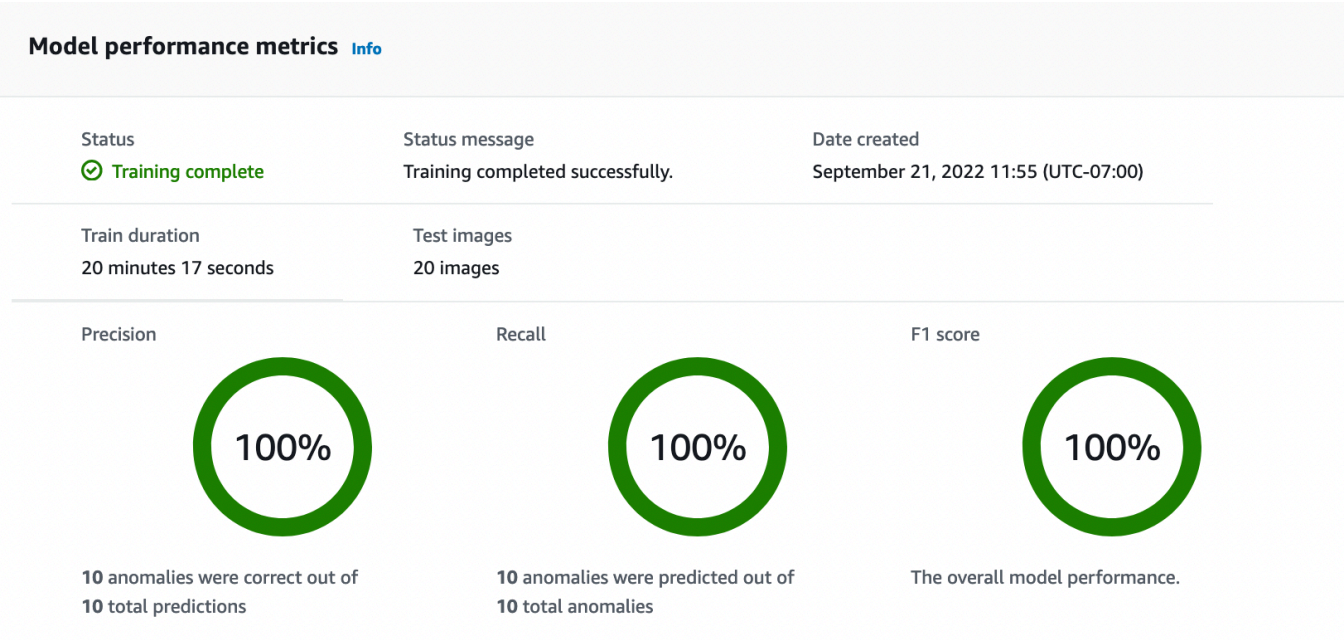
10. En la página de detalles de Entrenar modelo, elija Entrenar modelo.

11. En el cuadro de diálogo ¿Quiere entrenar su modelo?, escoja Entrenar modelo.
12. En la página de modelos del proyecto, puede ver que el entrenamiento ha comenzado. Para comprobar el estado actual, consulte la columna Estado correspondiente a la versión del modelo. El entrenamiento del modelo tarda al menos 30 minutos en completarse. El entrenamiento ha finalizado correctamente cuando el estado cambia a Entrenamiento completado.
13. Cuando finalice el entrenamiento, elija el modelo Modelo 1 en la página de modelos.

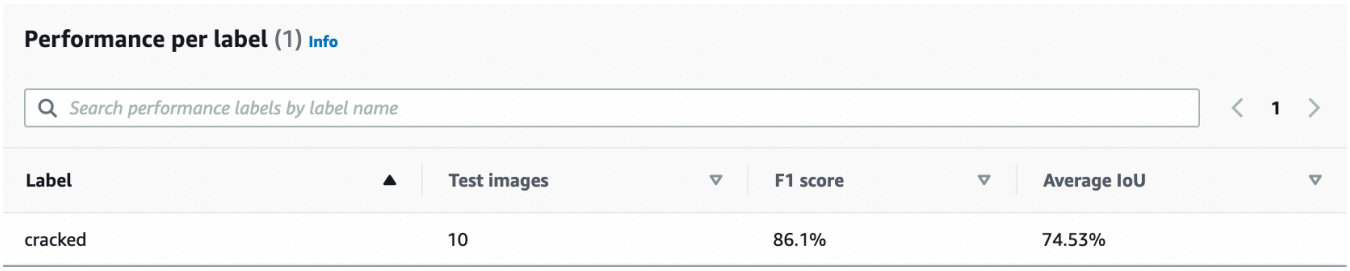


14. En la página de detalles del modelo, consulte los resultados de la evaluación en la pestaña Métricas de rendimiento. Las siguientes métricas están disponibles para lo siguiente:

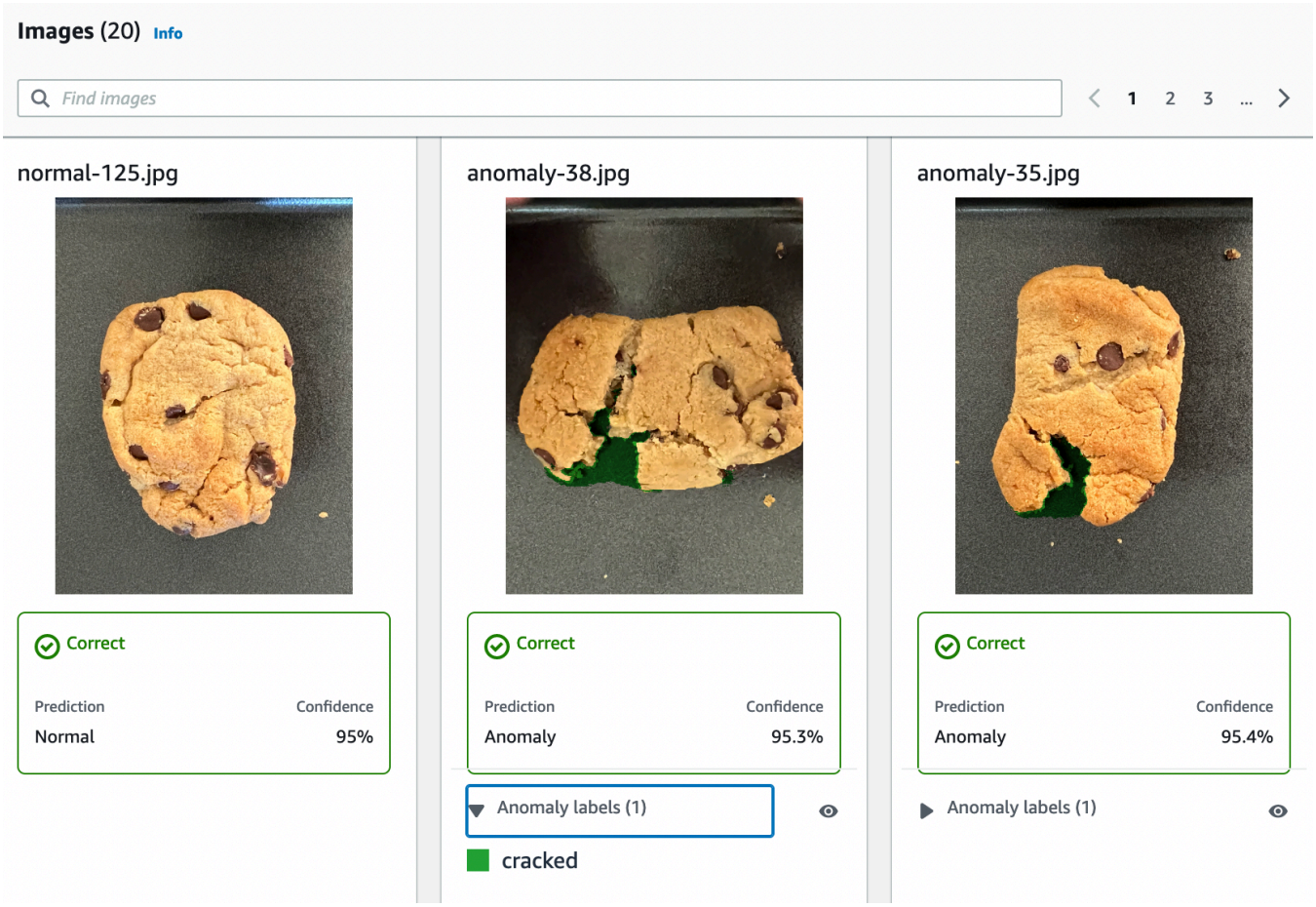
    - Métricas generales de rendimiento del modelo (precisión, recuperación y puntuación de F1) para las predicciones de clasificación realizadas por el modelo.



    - Métricas de rendimiento de las etiquetas de anomalías que se encuentran en las imágenes de prueba (IoU de media, puntuación de F1)



- Predicciones para las imágenes de prueba (clasificación, máscaras de segmentación y etiquetas de anomalías)



Como el entrenamiento de modelos no es determinista, los resultados de la evaluación pueden diferir de los que se muestran en esta página. Para obtener más información, consulte Mejorar su modelo Amazon Lookout for Vision.

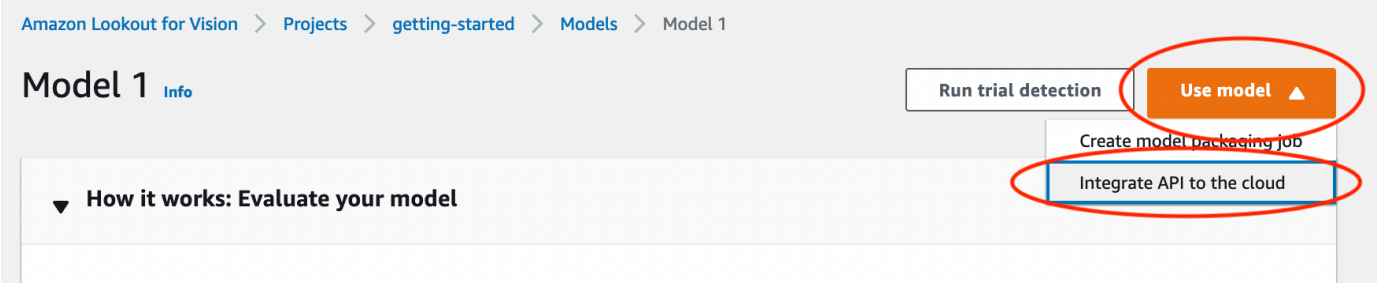
Paso 3: Iniciar el modelo

En este paso, comience a alojar el modelo para que esté listo para analizar las imágenes. Para obtener más información, consulte Ejecución de un modelo Amazon Lookout for Vision formado.

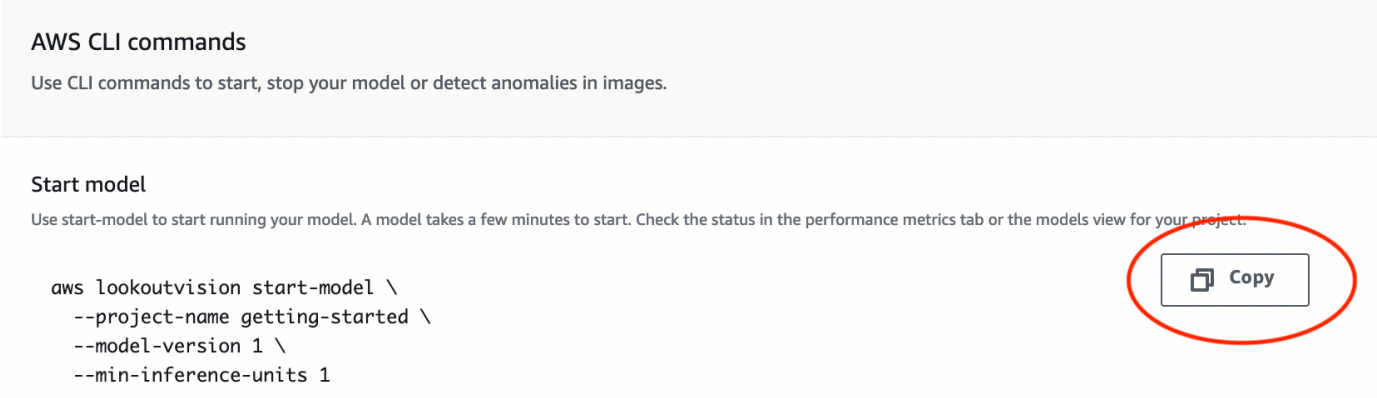
> **Note**
>
> Se le cobrará por la cantidad de tiempo en que se ejecute el modelo. Puede detener el modelo en Paso 5: Detener el modelo.

Iniciar el modelo.

1. En la página de detalles del modelo, seleccione Usar modelo y a continuación, seleccione Integrar la API en la nube.



2. En la sección de AWS CLI comandos, copia el `start-model` AWS CLI comando.



3. Asegúrese de que AWS CLI está configurado para ejecutarse en la misma AWS región en la que utiliza la consola Amazon Lookout for Vision. Para cambiar la AWS región que AWS CLI utilizan, consulteInstale los AWS SDK.

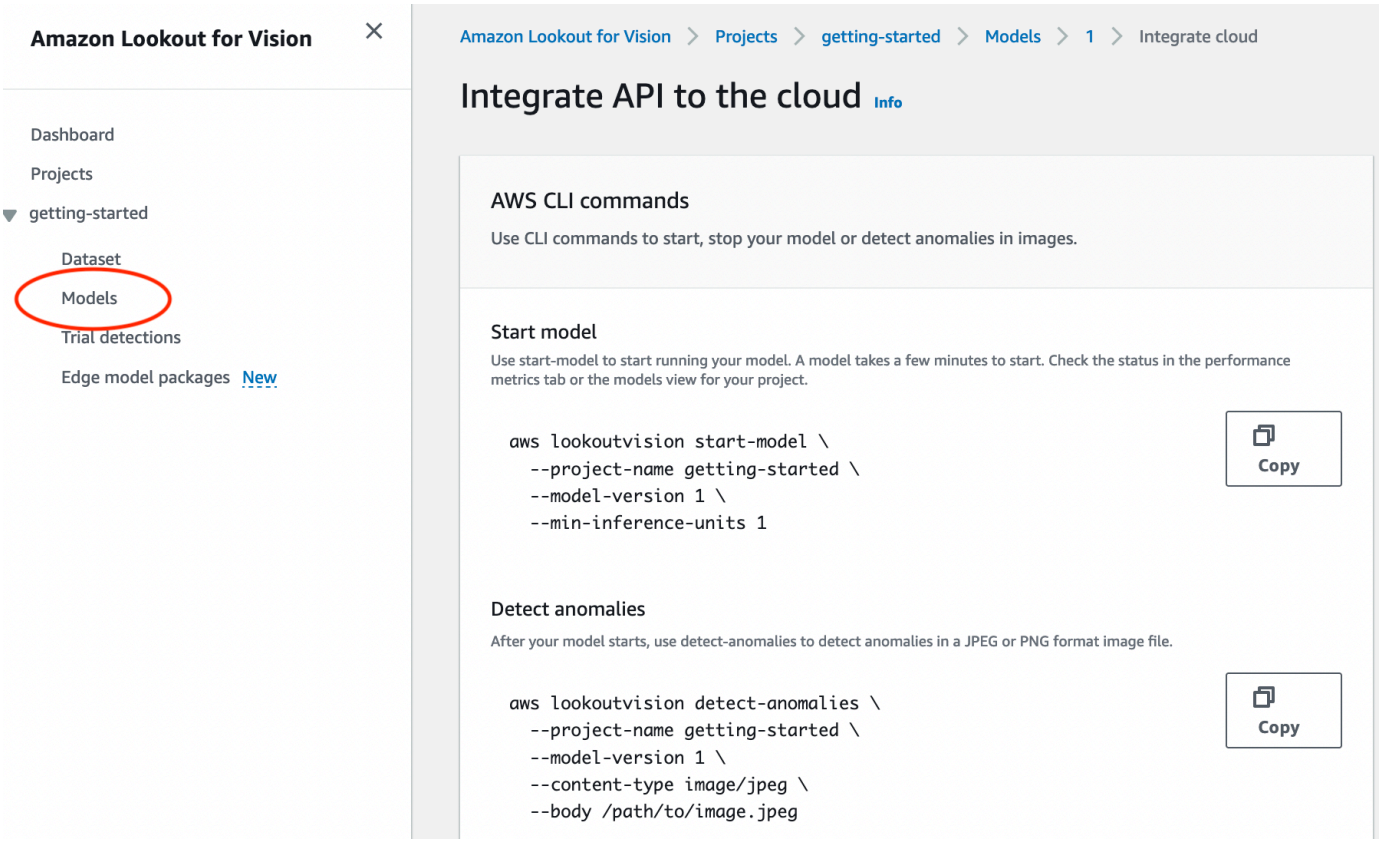
4. En la línea de comandos, inicie el modelo introduciendo el comando `start-model`. Si va a usar el perfil `lookoutvision` para obtener las credenciales, añada el parámetro `--profile lookoutvision-access`. Por ejemplo:

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

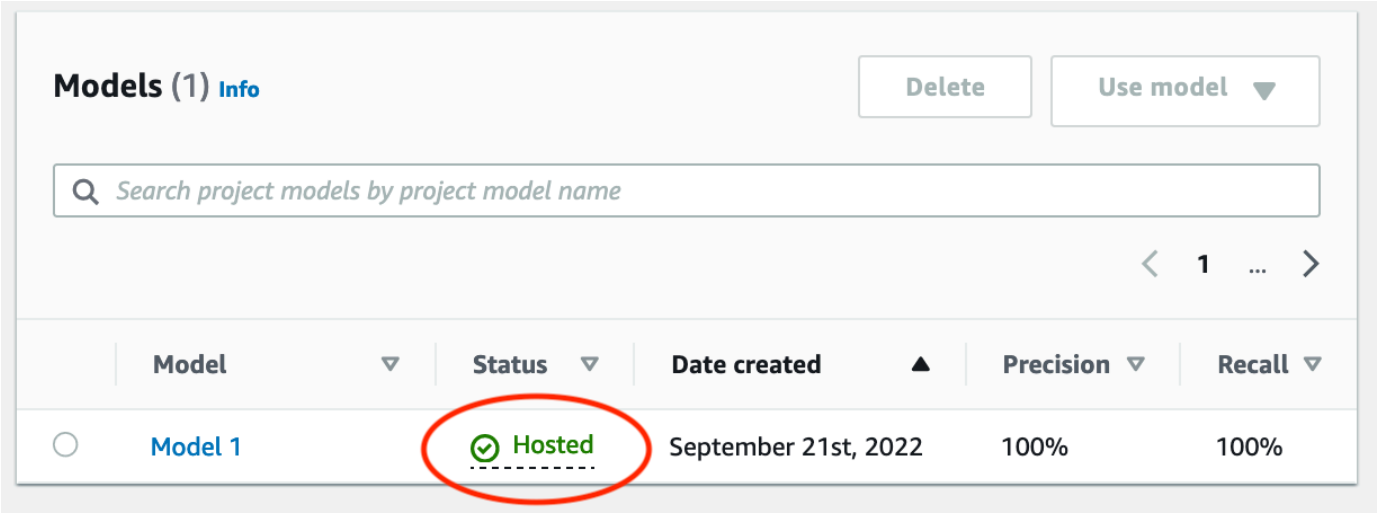
Si la llamada se realiza correctamente, se muestra el siguiente resultado:

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

5. En la consola, elija Modelos en la página de navegación.



6. Espere hasta que el estado del modelo (Modelo 1) de la columna Estado muestre Alojado. Si ya ha entrenado un modelo en el proyecto, espere a que se complete la última versión del modelo.

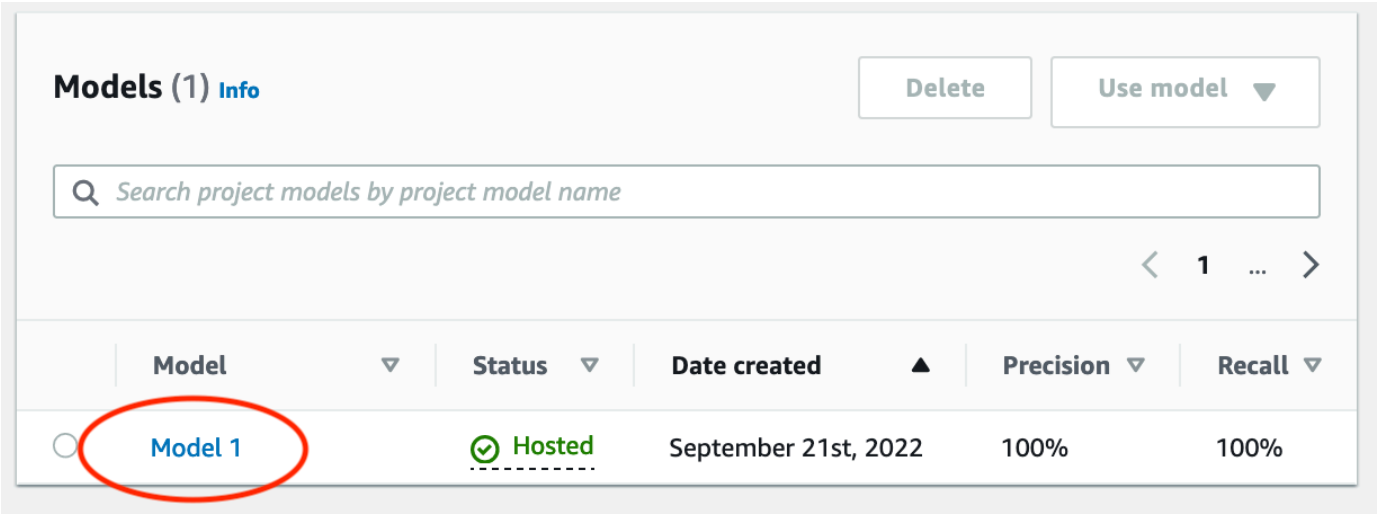


Paso 4: analice una imagen

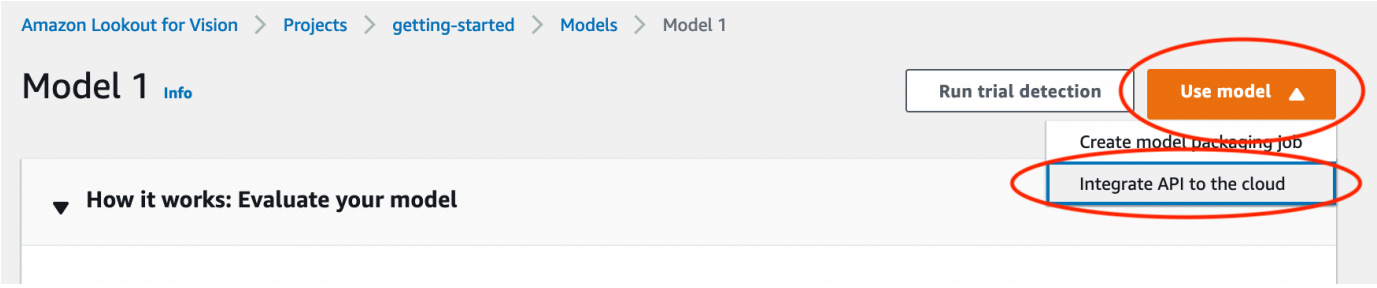
En este paso, analice una imagen con su modelo. Proporcionamos imágenes de ejemplo que puede utilizar en la carpeta `test-images` de introducción del repositorio de documentación de Lookout for Vision de su equipo. Para obtener más información, consulte Detección de anomalías en una imagen.

Analizar una imagen

1. En la página Modelos, elija el modelo Modelo 1.



2. En la página de detalles del modelo, seleccione Usar modelo y a continuación, seleccione Integrar la API en la nube.



3. En la sección de AWS CLI comandos, copie el `detect-anomalies` AWS CLI comando.

4. En la línea de comandos, analice una imagen anómala ingresando el `detect-anomalies` comando del paso anterior. Para el parámetro `--body`, especifique una imagen anómala de la carpeta `test-images` de introducción de su equipo. Si va a usar el perfil `lookoutvision` para obtener las credenciales, añada el parámetro `--profile lookoutvision-access`. Por ejemplo:

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

El resultado debería tener un aspecto similar al siguiente:

```
{
    "DetectAnomalyResult": {
        "Source": {
            "Type": "direct"
        },
        "IsAnomalous": true,
        "Confidence": 0.983975887298584,
        "Anomalies": [
            {
```

```
                "Name": "background",
                "PixelAnomaly": {
                    "TotalPercentageArea": 0.9818974137306213,
                    "Color": "#FFFFFF"
                }
            },
            {
                "Name": "cracked",
                "PixelAnomaly": {
                    "TotalPercentageArea": 0.018102575093507767,
                    "Color": "#23A436"
                }
            }
        ],
        "AnomalyMask": "iVBORw0KGgoAAAANSUhEUgAAAkAAAAMACA......"
    }
}
```

5. Tenga en cuenta lo siguiente sobre la salida:

   - `IsAnomalous` es un booleano para la clasificación prevista. `true` si la imagen es anómala; en caso contrario, `false`.
   - `Confidence`es un valor flotante que representa la confianza que Amazon Lookout for Vision tiene en la predicción. 0 es la confianza más baja y 1 es la confianza más alta.
   - `Anomalies`es una lista de las anomalías que se encuentran en la imagen. `Name`es la etiqueta de anomalía. `PixelAnomaly`incluye el área porcentual total de la anomalía (`TotalPercentageArea`) y un color (`Color`) para la etiqueta de la anomalía. La lista también incluye una anomalía de «fondo» que cubre el área que queda fuera de las anomalías que se encuentran en la imagen.
   - `AnomalyMask`es una imagen de máscara que muestra la ubicación de las anomalías en la imagen analizada.

   Puede utilizar la información de la respuesta para mostrar una combinación de la imagen analizada y la máscara de anomalías, tal como se muestra en el siguiente ejemplo. Para ver el código de ejemplo, consulte Mostrar información de clasificación y segmentación.


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

6. En la línea de comandos, analice una imagen normal de la `test-images` carpeta de introducción. Si va a usar el perfil `lookoutvision` para obtener las credenciales, añada el parámetro `--profile lookoutvision-access`. Por ejemplo:

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

   El resultado debería tener un aspecto similar al siguiente:

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

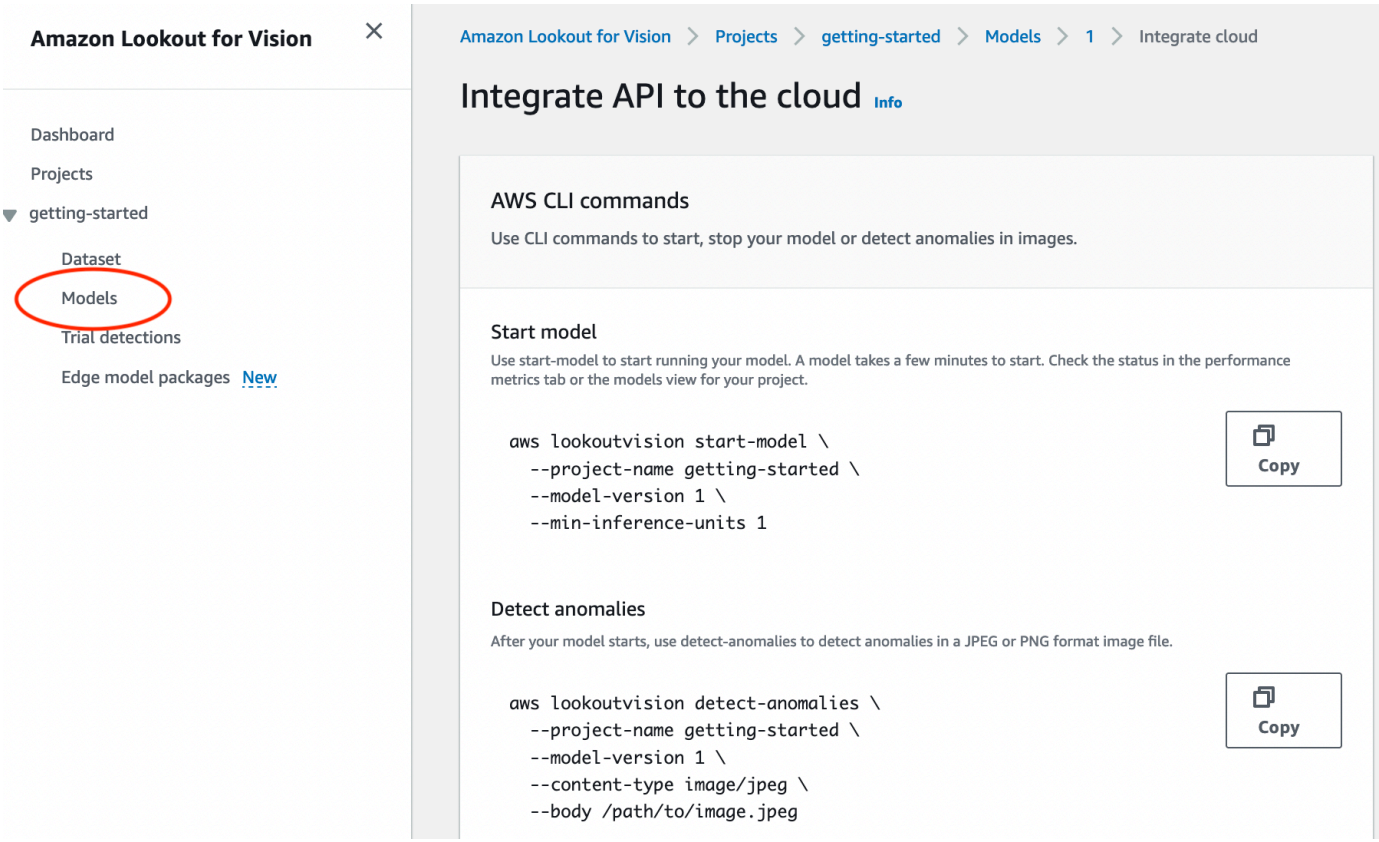
7. En el resultado, observe que el `false` valor de `IsAnomalous` clasifica la imagen como si no tuviera anomalías. `Confidence`Utilícelo para decidir su confianza en la clasificación. Además, la `Anomalies` matriz solo tiene la etiqueta de `background` anomalía.

Paso 5: Detener el modelo

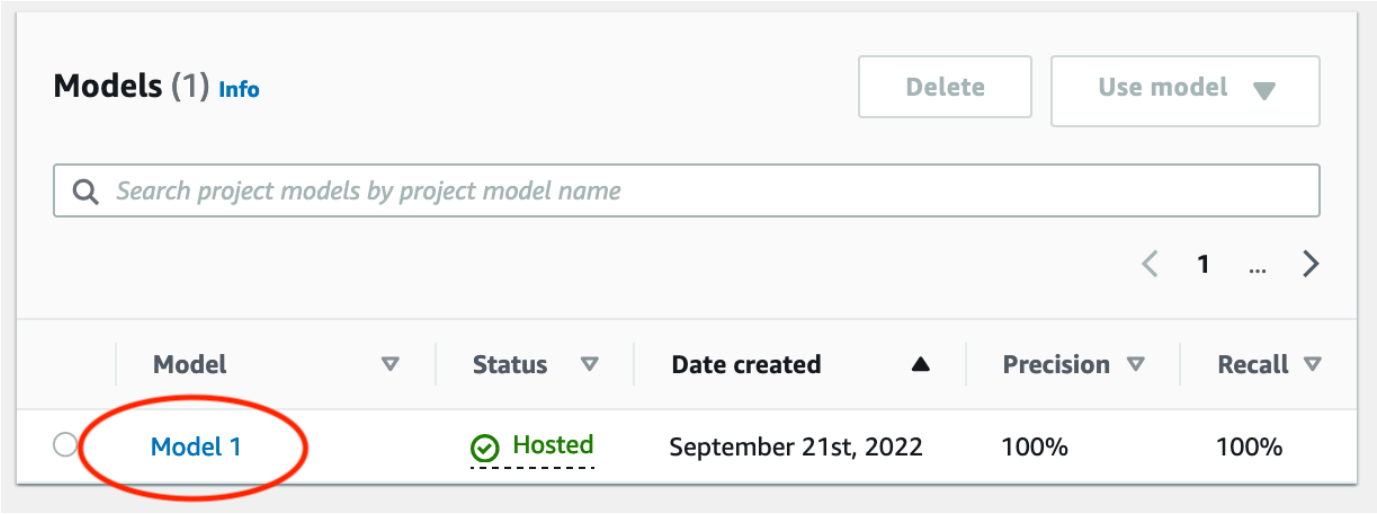
En este paso, dejará de alojar el modelo. Se le cobrará por la cantidad de tiempo de ejecución del modelo. Si no utiliza el modelo, debería detenerlo. Puede reiniciar el modelo la próxima vez que lo necesite. Para obtener más información, consulte Inicio del modelo Amazon Lookout for Vision.

Detener el modelo.

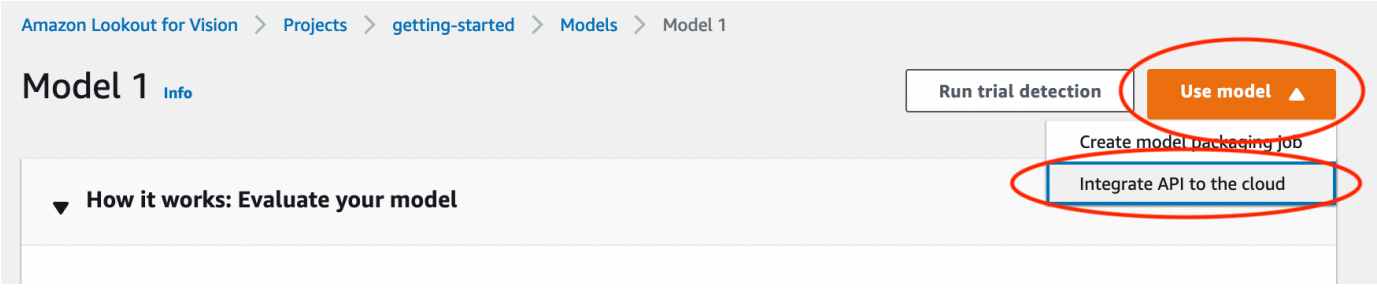
1. En el panel de navegación, elija Models.



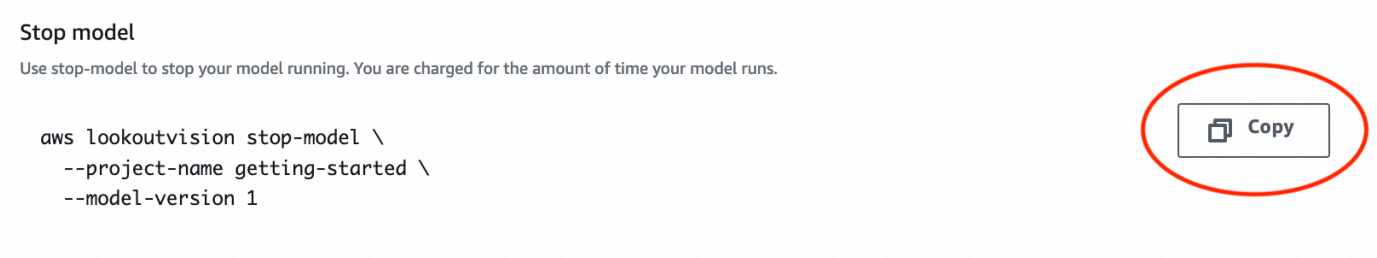
2. En la página Modelos, elija el modelo Modelo 1.



3. En la página de detalles del modelo, seleccione Usar modelo y a continuación, seleccione Integrar la API en la nube.



4. En la sección de AWS CLI comandos, copia el `stop-model` AWS CLI comando.



5. En la línea de comandos, detenga el modelo introduciendo el `stop-model` AWS CLI comando del paso anterior. Si va a usar el perfil `lookoutvision` para obtener las credenciales, añada el parámetro `--profile lookoutvision-access`. Por ejemplo:

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

Si la llamada se realiza correctamente, se muestra el siguiente resultado:

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

6. En la consola, elija Modelos en la página de navegación izquierda.
7. El modelo se ha detenido cuando el valor de la columna Estado es Entrenamiento finalizado.

Pasos a seguir a continuación

Cuando esté listo para crear un modelo con sus propias imágenes, comience por seguir las instrucciones que se indican en Creación de su proyecto. Las instrucciones incluyen los pasos para crear un modelo con la consola Amazon Lookout for Vision y con AWS el SDK.

Si quiere probar otros conjuntos de datos de ejemplo, consulte Ejemplos de código y conjuntos de datos.

Creación de Amazon Lookout for Vision

Un modelo de Amazon Lookout for Vision es un modelo de machine learning que predice la presencia de anomalías en imágenes nuevas mediante la búsqueda de patrones en las imágenes utilizadas para entrenar el modelo. En esta sección le mostramos cómo crear y entrenar un modelo. Después de entrenar un modelo, debe evaluar su rendimiento. Para obtener más información, consulte Mejorar su modelo Amazon Lookout for Vision.

Antes de crear su primer modelo, le recomendamos que lea y. Análisis Amazon Lookout for Vision Introducción a Amazon Lookout for Vision Si está utilizando el AWS SDK, lea Llame a una operación de Amazon Lookout for Vision.

Temas

- Creación de su proyecto
- Creación de un conjunto de datos
- Etiquetado de imágenes
- Entrenamiento del modelo
- Solución de problemas en el entrenamiento de modelos

Creación de su proyecto

Un proyecto de Amazon Lookout for Vision es una agrupación de los recursos necesarios para crear y administrar un modelo de Lookout for Vision. El proyecto administra lo siguiente:

- Conjuntos de datos: las imágenes y las etiquetas de imagen que se utilizan para entrenar un modelo. Para obtener más información, consulte Creación de un conjunto de datos.
- Modelo: el software que se entrena para detectar anomalías. Para tener varias versiones, puede tener varias versiones. Para obtener más información, consulte Entrenamiento del modelo.

Le recomendamos que utilice un proyecto para un único caso de uso, como la detección de anomalías en un único tipo de pieza de la máquina.

> **Note**
>
> Se puede utilizar AWS CloudFormation para aprovisionar y configurar proyectos de Amazon Lookout for Vision. Para obtener más información, consulte Creación de recursos Amazon Lookout for Vision con AWS CloudFormation.

Para ver sus proyectos, consulte Visualización de sus proyectos o abra Uso del panel de Amazon Lookout for Vision. Para eliminar un modelo, consulte Eliminación de un modelo.

Temas

- Creación de un proyecto (consola)
- Creación de un proyecto (SDK)

Creación de un proyecto (consola)

En el siguiente procedimiento se explica cómo crear un proyecto con la consola.

Cómo crear un proyecto (consola)

1. Abre la consola https://console.aws.amazon.com/lookoutvision/Amazon Lookout for Vision en.
2. En el panel de navegación izquierdo, elija Proyectos.
3. Elija Crear proyecto.
4. En Nombre del proyecto, asigne un nombre al proyecto.
5. Elija Crear proyecto. Se abrirá la página de detalles del proyecto.
6. Siga los pasos que se indican en Creación de un conjunto de datos para crear un conjunto de datos.

Creación de un proyecto (SDK)

La CreateProjectoperación se utiliza para crear un proyecto de Amazon Lookout for Vision. La respuesta de la `CreateProject` incluye el nombre de recurso de Amazon (ARN) del proyecto. Después, llame CreateDatasetpara añadir un conjunto de datos de formación y uno de prueba a su proyecto. Para obtener más información, consulte Creación de un conjunto de datos con un archivo de manifiesto (SDK).

Para ver los proyectos que ha creado en un proyecto, llame `ListProjects`. Para obtener más información, consulte Visualización de sus proyectos.

Para crear 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: Configura el AWS CLI y AWS SDKs.
2. Utilice el siguiente ejemplo de código para eliminar un modelo.

   CLI

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

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

   Python

   Este código se ha tomado del GitHub repositorio de ejemplos del SDK de AWS documentación. Consulte el ejemplo completo aquí.

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

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

Java V2

Este código se ha tomado del GitHub repositorio de ejemplos del SDK de AWS documentación. Consulte el ejemplo completo aquí.

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

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

        CreateProjectResponse response =
 lfvClient.createProject(createProjectRequest);

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

        return response.projectMetadata();

}
```

3. Siga los pasos que se indican en Creación de un conjunto de datos mediante un archivo de manifiesto de Amazon SageMaker AI Ground Truth para crear un conjunto de datos.

Creación de un conjunto de datos

Los conjuntos de datos contienen las imágenes y las etiquetas asignadas que se utilizan para entrenar y probar un modelo. El conjunto de datos para su proyecto se crea con la consola Amazon Lookout for Vision o con CreateDatasetla operación. Las imágenes del conjunto de datos deben

etiquetarse de acuerdo con el tipo de modelo que desee crear (clasificación o segmentación de imágenes).

Temas

Preparación de imágenes para un conjunto de datos

Necesita una colección de imágenes para crear un conjunto de datos. Las imágenes deben estar en formato PNG o JPEG. La cantidad y el tipo de imágenes que necesita dependen de si su proyecto tiene un único conjunto de datos o conjuntos de datos de entrenamiento y prueba independientes.

Proyecto con un conjunto de datos individual

Para crear un modelo de clasificación de imágenes, necesita lo siguiente para empezar a entrenar:

- Al menos 20 imágenes de objetos normales.
- Al menos 10 imágenes de objetos anómalos.

Para crear un modelo de segmentación de imágenes, necesita lo siguiente para empezar a entrenar:

- Al menos 20 imágenes de cada tipo de anomalía.
- Cada imagen anómala (imagen con tipos de anomalías presentes) debe tener solo un tipo de anomalía.
- Al menos 20 imágenes de objetos normales.

Proyecto de conjunto de datos de entrenamiento y prueba independientes

Para crear un modelo de clasificación de imágenes, necesita lo siguiente:

- Al menos 10 imágenes de objetos normales en el conjunto de datos de entrenamiento.

- Al menos 10 imágenes de objetos normales en el conjunto de datos de prueba.
- Al menos 10 imágenes de objetos anómalos en el conjunto de datos de prueba.

Para crear un modelo de segmentación de imágenes, necesita lo siguiente:

- Cada conjunto de datos necesita al menos 10 imágenes de cada tipo de anomalía.
- Cada imagen anómala (imagen con tipos de anomalías presentes) debe tener solo un tipo de anomalía.
- Cada conjunto de datos debe tener al menos 10 imágenes de objetos normales.

Para crear un modelo de mayor calidad, utilice más imágenes que el número mínimo. Si va a crear un modelo de segmentación, recomendamos incluir imágenes con varios tipos de anomalías, pero estas no cuentan para el mínimo que Lookout for Vision necesita para empezar a entrenar.

Las imágenes deben ser de un solo tipo de objeto. Además, debe disponer de condiciones de captura de imágenes uniformes, como la posición de la cámara, la iluminación y la postura del objeto.

Todas las imágenes de los conjuntos de datos de entrenamiento y prueba deben tener las mismas dimensiones. Más adelante, las imágenes que analice con su modelo entrenado deben tener las mismas dimensiones que las imágenes del conjunto de datos de entrenamiento y prueba. Para obtener más información, consulte Detección de anomalías en una imagen.

Todas las imágenes de entrenamiento y prueba deben ser imágenes únicas, preferiblemente de objetos únicos. Las imágenes normales deben capturar las variaciones normales del objeto que se está analizando. Las imágenes anómalas deben capturar una muestra diversa de anomalías.

Amazon Lookout for Vision proporciona imágenes de ejemplo que puede utilizar. Para obtener más información, consulte Conjunto de datos de clasificación de imágenes.

Para conocer los límites de imágenes, consulte Cuotas.

Creación del conjunto de datos

Cuando cree el conjunto de datos para su proyecto, elija la configuración inicial del conjunto de datos de su proyecto. También elige desde dónde importa Lookout for Vision las imágenes.

Elegir una configuración de conjunto de datos para su proyecto

Al crear el primer conjunto de datos de su proyecto, debe elegir una de las siguientes configuraciones de conjunto de datos:

- Conjunto de datos único: un proyecto de conjunto de datos único utiliza un único conjunto de datos para entrenar y probar el modelo. El uso de un único conjunto de datos simplifica el entrenamiento al permitir que Amazon Lookout for Vision elija las imágenes de entrenamiento y prueba. Durante el entrenamiento, Amazon Lookout for Vision divide internamente el conjunto de datos en un conjunto de datos de entrenamiento y un conjunto de datos de prueba. No tiene acceso a este conjunto de datos dividido. Recomendamos utilizar un proyecto de conjunto de datos único para la mayoría de los escenarios.
- Separar conjuntos de datos de entrenamiento y prueba: si quiere tener un control más preciso sobre el entrenamiento, las pruebas y el ajuste del rendimiento, puede configurar su proyecto para que tenga conjuntos de datos de entrenamiento y prueba independientes. Use un conjunto de datos de prueba independiente si quiere controlar las imágenes que se utilizan para las pruebas o si ya tiene un conjunto de imágenes de referencia que desee usar.

Puede añadir un conjunto de datos de entrenamiento o un conjunto de datos de prueba a un proyecto existente. El conjunto de datos único se convierte entonces en el conjunto de datos de entrenamiento. Si elimina el conjunto de datos de prueba de un proyecto con conjuntos de datos de entrenamiento y de prueba independientes, el proyecto se convierte en un proyecto de conjunto de datos único. Para obtener más información, consulte Eliminación de un conjunto de datos.

Importación de imágenes

Al crear un conjunto de datos, elija desde dónde importar las imágenes. Según cómo importe las imágenes, es posible que las imágenes también estén etiquetadas. Si las imágenes no se etiquetan después de crear el conjunto de datos, consulte Etiquetado de imágenes.

Puede crear un conjunto de datos e importar sus imágenes de una de las siguientes maneras:

- Importar imágenes de un equipo local. Las imágenes no están etiquetadas. Puede añadir o etiquetar con la consola de Lookout for Vision.
- Importar imágenes de un bucket de S3. Amazon Lookout for Vision puede clasificar las imágenes mediante los nombres de las carpetas para etiquetarlas. Use `normal` para imágenes normales. Se utiliza `anomaly` para imágenes anómalas. No puede asignar etiquetas de segmentación automáticamente.

- Importa un archivo de manifiesto de Amazon SageMaker AI Ground Truth, que incluye imágenes etiquetadas. Puede crear e importar su propio archivo de manifiesto. Si tiene muchas imágenes, considere utilizar el servicio de etiquetado SageMaker AI Ground Truth. A continuación, importas el archivo de manifiesto de salida del trabajo de Amazon SageMaker AI Ground Truth. Si lo necesita, puede utilizar la consola de Lookout for Vision para añadir o cambiar etiquetas.

Si utilizas el AWS SDK, creas un conjunto de datos con un archivo de manifiesto de Amazon SageMaker AI Ground Truth. Para obtener más información, consulte Creación de un conjunto de datos mediante un archivo de manifiesto de Amazon SageMaker AI Ground Truth.

Si, después de crear el conjunto de datos, se etiquetan las imágenes, puede entrenar el modelo. Si las imágenes no están etiquetadas, añada las etiquetas según el tipo de modelo que desee crear. Para obtener más información, consulte Etiquetado de imágenes.

Puede añadir más imágenes a un conjunto de datos existente. Para obtener más información, consulte Añadir imágenes a su conjunto de datos..

Creación de un conjunto de datos con imágenes almacenadas en su equipo local

Puede crear un conjunto de datos mediante imágenes que se cargan directamente desde el equipo. Puede cargar hasta 30 imágenes a la vez. En este procedimiento, puede crear un único proyecto de conjunto de datos o un proyecto con conjuntos de datos de entrenamiento y prueba independientes.

> **Note**
>
> Si acaba de completar Creación de su proyecto, la consola debería mostrar el panel de control del proyecto y no es necesario que realices los pasos del 1 al 3.

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

1. Abre la consola https://console.aws.amazon.com/lookoutvision/Amazon Lookout for Vision en.
2. En el panel de navegación izquierdo, elija Proyectos.
3. En la página Proyectos, elija el proyecto al que desee añadir el conjunto de datos.
4. En la página de detalles del proyecto, elija Crear conjunto de datos.

5. Seleccione la pestaña Conjunto de datos único o la pestaña Separar conjuntos de datos de entrenamiento y prueba y siga los pasos.

   Single dataset

   a. En la sección Configuración del conjunto de datos, elija Crear un solo conjunto de datos.
   b. En la sección de configuración de la fuente de la imagen, seleccione Cargar imágenes desde el equipo.
   c. Elija Crear conjunto de datos.
   d. En la página del conjunto de datos del proyecto, seleccione Agregar imágenes.
   e. Escoja las imágenes que quiera cargar en el conjunto de datos de los archivos del equipo. Puede arrastrar las imágenes o elegir las imágenes deseadas de su equipo local.
   f. Seleccione Cargar imágenes.

   Separate training and test datasets

   a. En la sección Configuración del conjunto de datos, elija Crear un conjunto de datos de entrenamiento y un conjunto de datos de prueba.
   b. En la sección Detalles del conjunto de datos de entrenamiento, seleccione Cargar imágenes desde el equipo.
   c. 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.

   d. Elija Crear conjunto de datos. Se abrirá la página de conjuntos de datos del proyecto con las pestañas Entrenamiento y Prueba con los conjuntos de datos respectivos.
   e. Seleccione Acciones y luego Agregar imágenes al conjunto de datos de entrenamiento.
   f. Escoja las imágenes que quiera cargar en el conjunto de datos. Puede arrastrar las imágenes o elegir las imágenes deseadas de su equipo local.
   g. Seleccione Cargar imágenes.

h. Repita los pasos del 5e al 5g. En el paso 5e, seleccione Acciones y luego Agregar imágenes al conjunto de datos de entrenamiento.

6. Siga los pasos que se indican en Etiquetado de imágenes para etiquetar las imágenes.
7. Siga los pasos que se indican en Entrenamiento del modelo para entrenar su modelo.

Creación de un conjunto de datos con imágenes almacenadas en un bucket de Amazon S3

Puede crear un conjunto de datos utilizando imágenes almacenadas en un bucket de Amazon S3. Con esta opción, puede utilizar la estructura de carpetas de su bucket de Amazon S3 para clasificar automáticamente las imágenes. Puede almacenar las imágenes en el bucket de la consola o en otro bucket de Amazon S3 de su cuenta.

Configurar carpetas para el etiquetado automático

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 de la ruta de carpeta de Amazon S3 que especifique en el URI de S3 al crear el conjunto de datos.

La siguiente es la `train` carpeta de las imágenes de ejemplo de introducción. Si especifica la ubicación de la carpeta Amazon S3 como `S3-bucket/circuitboard/train/`, a las imágenes de la carpeta `normal` se les asigna la etiqueta `Normal`. A las imágenes de la carpeta `anomaly` se les asigna la etiqueta `Anomaly`. Los nombres de las carpetas secundarias más profundas no se utilizan para etiquetar las imágenes.

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

```
        ### .
```

Creación de un conjunto de datos con imágenes de un bucket de Amazon S3

En el siguiente procedimiento, se crea un conjunto de datos utilizando imágenes de ejemplo de clasificación almacenadas en un bucket de Amazon S3. Para usar sus propias imágenes, cree la estructura de carpetas que se describe en Configurar carpetas para el etiquetado automático.

El procedimiento también muestra cómo crear un proyecto de conjunto de datos único o un proyecto que utilice conjuntos de datos de entrenamiento y prueba independientes.

Si no elige etiquetar automáticamente las imágenes, tendrá que etiquetarlas después de crear los conjuntos de datos. Para obtener más información, consulte Clasificación de imágenes (consola).

> **Note**
>
> Si acaba de completar Creación de su proyecto, la consola debería mostrar el panel de control del proyecto y no es necesario que realice los pasos del 1 al 4.

Para crear un conjunto de datos con imágenes almacenadas en un bucket de Amazon S3

1. Si aún no lo ha hecho, cargue las imágenes de introducción en su bucket de Amazon S3. Para obtener más información, consulte Conjunto de datos de clasificación de imágenes.
2. Abre la consola https://console.aws.amazon.com/lookoutvision/Amazon Lookout for Vision en.
3. En el panel de navegación izquierdo, elija Proyectos.
4. En la página Proyectos, elija el proyecto al que desee añadir el conjunto de datos. Se abrirá la página de detalles del proyecto.
5. Elija Crear conjunto de datos. Se muestra la página Crear conjunto de datos.

   > **Tip**
   >
   > Si sigue las instrucciones de introducción, seleccione Crear un conjunto de datos de entrenamiento y un conjunto de datos de prueba.

6. Seleccione la pestaña Conjunto de datos único o la pestaña Separar conjuntos de datos de entrenamiento y prueba y siga los pasos.

Single dataset

a. En la sección Configuración del conjunto de datos, elija Crear un solo conjunto de datos.
b. Introduzca la información de los pasos del 7 al 9 en la sección Configuración de la fuente de la imagen.

Separate training and test datasets

a. En la sección Configuración del conjunto de datos, elija Crear un conjunto de datos de entrenamiento y un conjunto de datos de prueba.
b. Para su conjunto de datos de entrenamiento, introduzca la información de los pasos del 7 al 9 en la sección Detalles del conjunto de datos de entrenamiento.
c. Para su conjunto de datos de prueba, introduzca la información de los pasos del 7 al 9 en la sección Detalles del conjunto de datos de prueba.

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

7. Seleccione Importar imágenes del bucket de Amazon S3.
8. En el URI de S3, introduzca la ubicación del bucket de Amazon S3 y la ruta de la carpeta. Cambie `bucket` por el nombre de su bucket de Amazon S3.

   a. Si va a crear un proyecto de conjunto de datos único o un conjunto de datos de entrenamiento, introduzca lo siguiente:

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

   b. Si va a crear un conjunto de datos de prueba, introduzca lo siguiente:

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

9. Elija Adjuntar etiquetas automáticamente a las imágenes en función de la carpeta.
10. Elija Crear conjunto de datos. Se abre una página de conjunto de datos con las imágenes etiquetadas.
11. Siga los pasos que se indican en Entrenamiento del modelo para entrenar su modelo.

Creación de un conjunto de datos mediante un archivo de manifiesto de Amazon SageMaker AI Ground Truth

Un archivo de manifiesto contiene información sobre las imágenes y las etiquetas de las imágenes que puede usar para entrenar y probar un modelo. Puede almacenar un archivo de manifiesto en un bucket de Amazon S3 y utilizarlo para crear un conjunto de datos. Puedes crear tu propio archivo de manifiesto o utilizar un archivo de manifiesto existente, como el resultado de un trabajo de Amazon SageMaker AI Ground Truth.

Temas

Cómo usar un trabajo de Amazon SageMaker Ground Truth

El etiquetado de las imágenes puede llevar bastante tiempo. Por ejemplo, dibujar con precisión una máscara alrededor de una anomalía puede tardar decenas de segundos. Si tiene cientos de imágenes, es posible que tarde varias horas en etiquetarlas. Como alternativa a etiquetar las imágenes tú mismo, considera usar Amazon SageMaker Ground Truth.

Con Amazon SageMaker AI Ground Truth, puede utilizar trabajadores de Amazon Mechanical Turk, una empresa proveedora que elija, o de personal interno privado para crear un conjunto de imágenes etiquetadas. Para obtener más información, consulte Uso de Amazon SageMaker AI Ground Truth para etiquetar datos.

El uso de Amazon Mechanical Turk conlleva un coste. Además, completar un trabajo de etiquetado de Amazon Ground Truth puede llevar varios días. Si el coste es un problema o si necesita entrenar su modelo rápidamente, le recomendamos que utilice la consola de Amazon Lookout for Vision para etiquetar sus imágenes.

Puede utilizar un trabajo de etiquetado de Amazon SageMaker AI Ground Truth para etiquetar las imágenes de forma adecuada para los modelos de clasificación de imágenes y los modelos de segmentación de imágenes. Una vez finalizado el trabajo, se utiliza el archivo de manifiesto de salida para crear un conjunto de datos de Amazon Lookout for Vision.

Clasificación de imágenes

Para etiquetar imágenes para un modelo de clasificación de imágenes, cree un trabajo de etiquetado para una tarea de clasificación de imágenes (etiqueta única).

Segmentación de imágenes

Para etiquetar imágenes para un modelo de segmentación de imágenes, cree un trabajo de etiquetado para una tarea de clasificación de imágenes (etiqueta única). A continuación, encadene el trabajo para crear un trabajo de etiquetado para una tarea de segmentación semántica de imágenes.

También puede usar un trabajo de etiquetado para crear un archivo de manifiesto parcial para un modelo de segmentación de imágenes. Por ejemplo, puede clasificar las imágenes con una tarea de clasificación de imágenes (etiqueta única). Tras crear un conjunto de datos de Lookout for Vision con el resultado del trabajo, utilice la consola de Amazon Lookout for Vision para añadir máscaras de segmentación y etiquetas de anomalías a las imágenes del conjunto de datos.

Etiquetado de imágenes con Amazon SageMaker AI Ground Truth

El siguiente procedimiento muestra cómo etiquetar imágenes con las tareas de etiquetado de imágenes de Amazon SageMaker AI Ground Truth. El procedimiento crea un archivo de manifiesto de clasificación de imágenes y, si lo desea, encadena la tarea de etiquetado de imágenes para crear un archivo de manifiesto de segmentación de imágenes. Si desea que su proyecto tenga un conjunto de datos de prueba independiente, repita este procedimiento para crear el archivo de manifiesto para el conjunto de datos de prueba.

Para etiquetar imágenes con Amazon SageMaker AI Ground Truth (consola)

1. Cree un trabajo de Ground Truth para una tarea de clasificación de imágenes (etiqueta única) siguiendo las instrucciones de Create a Labeling Job (consola).

   a. En el paso 10, elija Imagen en el menú desplegable Categoría de tareas y elija Clasificación de imágenes (etiqueta única) como tipo de tarea.

   b. En el paso 16, en la sección de la herramienta de etiquetado de clasificación de imágenes (etiqueta única), añada dos etiquetas: normal y anormal.

2. Espere a que el personal termine de clasificar sus imágenes.

3. Si va a crear un conjunto de datos para un modelo de segmentación de imágenes, haga lo siguiente. Continúe con el paso 4.

   a. En la consola Amazon SageMaker AI Ground Truth, abre la página de trabajos de etiquetado.

   b. Elija el trabajo que ha creado anteriormente. Esto habilita el menú Acciones.

   c. En el menú Acciones, seleccione Encadenar. Se abrirá la página de detalles del trabajo.

   d. En el tipo de tarea, elija la segmentación semántica.
   e. Elija Siguiente.
   f. En la sección Herramienta de etiquetado para segmentación semántica, añada etiquetas de anomalías para cada tipo de anomalía que desee que encuentre su modelo.
   g. Seleccione Crear.
   h. Espere a que el personal etiquete sus imágenes.
4. Abra la consola Ground Truth y abra la página de trabajos de etiquetado.
5. Si va a crear un modelo de clasificación de imágenes, elija el trabajo que ha creado en el paso 1. Si va a crear un modelo de segmentación de imágenes, elija el trabajo creado en el paso 3.
6. En el resumen del trabajo de etiquetado, abra la ubicación S3 en la ubicación del conjunto de datos de salida. Anote la ubicación del archivo de manifiesto, que debería ser `s3://output-dataset-location/manifests/output/output.manifest`.
7. Repita este procedimiento si desea crear un archivo de manifiesto para un conjunto de datos de prueba. De lo contrario, siga las instrucciones que aparecen en Creación del conjunto de datos para crear un conjunto de datos con el archivo de manifiesto.

Creación del conjunto de datos

Utilice este procedimiento para crear un conjunto de datos en un proyecto de Lookout for Vision con el archivo de manifiesto que indicó en el paso 6 de Etiquetado de imágenes con Amazon SageMaker AI Ground Truth. El archivo de manifiesto crea el conjunto de datos de entrenamiento para un proyecto de conjunto de datos único. Si quieres que tu proyecto tenga un conjunto de datos de prueba independiente, puedes ejecutar otro trabajo de Amazon SageMaker AI Ground Truth para crear un archivo de manifiesto para el conjunto de datos de prueba. O bien, puede crear el archivo de manifiesto usted mismo. También puede importar imágenes a su conjunto de datos de prueba desde un bucket de Amazon S3 o desde su equipo local. (Es posible que sea necesario etiquetar las imágenes antes de poder entrenar el modelo).

En este procedimiento se presupone que el proyecto no tiene ningún conjunto de datos.

Para crear un conjunto de datos con Lookout for Vision (consola)

1. Abre la consola https://console.aws.amazon.com/lookoutvision/Amazon Lookout for Vision en.
2. Elija Comenzar.
3. En el panel de navegación izquierdo, elija Proyectos.

4. Elija el proyecto que desea agregar para usarlo con el archivo de manifiesto.
5. En la sección Cómo funciona, seleccione Crear conjunto de datos.
6. Seleccione la pestaña Conjunto de datos único o la pestaña Separar conjuntos de datos de entrenamiento y prueba y siga los pasos.

   Single dataset

   1. Elija Crear un solo conjunto de datos.
   2. En la sección Configuración de la fuente de la imagen, selecciona Importar imágenes etiquetadas por SageMaker Ground Truth.
   3. Para la ubicación del archivo. manifest, introduzca la ubicación del archivo de manifiesto que anotó en el paso 6 de Etiquetado de imágenes con Amazon SageMaker AI Ground Truth.

   Separate training and test datasets

   1. Elija Crear un conjunto de datos de entrenamiento y un conjunto de datos de prueba.
   2. En la sección de detalles del conjunto de datos de entrenamiento, selecciona Importar imágenes etiquetadas por SageMaker Ground Truth.
   3. En la ubicación del archivo. manifest, la ubicación del archivo de manifiesto que anotó en el paso 6 de Etiquetado de imágenes con Amazon SageMaker AI Ground Truth.
   4. En la sección Detalles del conjunto de datos de prueba, selecciona Importar imágenes etiquetadas por SageMaker Ground Truth.
   5. En la ubicación del archivo. manifest, la ubicación del archivo de manifiesto que anotó en el paso 6 de Etiquetado de imágenes con Amazon SageMaker AI Ground Truth. Recuerde que necesita un archivo de manifiesto independiente para el conjunto de datos de prueba.
7. Elija Enviar.
8. Siga los pasos que se indican en Entrenamiento del modelo para entrenar su modelo.

Creación de un archivo de manifiesto

Puedes crear un conjunto de datos 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. Existen diferentes formatos para la clasificación y segmentación de imágenes. Los archivos de manifiesto deben ser codificados usando la codificación UTF-8.

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

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 en un bucket diferente. Debe especificar la ubicación de una imagen en el `source-ref` campo de una línea JSON.

Puede crear un archivo de manifiesto mediante código. El cuaderno de Python de Amazon Lookout for Vision Lab muestra cómo crear un archivo de manifiesto de clasificación de imágenes para las imágenes de ejemplo de la placa de circuito. Como alternativa, puedes usar el código de ejemplo de los conjuntos de datos del repositorio de ejemplos AWS de código. Puede crear fácilmente un archivo de manifiesto mediante un archivo de valores separados por comas (CSV). Para obtener más información, consulte Creación de un archivo de manifiesto de clasificación a partir de un archivo CSV.

Temas

- Definir líneas JSON para la clasificación de imágenes
- Definir líneas JSON para la segmentación de imágenes
- Creación de un archivo de manifiesto de clasificación a partir de un archivo CSV
- Creación de un conjunto de datos con un archivo de manifiesto (consola)
- Creación de un conjunto de datos con un archivo de manifiesto (SDK)

Definir líneas JSON para la clasificación de imágenes

Defina una línea JSON para cada imagen que desee utilizar en un archivo de manifiesto de Amazon Lookout for Vision. Si desea crear un modelo de clasificación, la línea JSON debe incluir una

clasificación de imágenes que sea normal o anómala. Una línea JSON está en formato SageMaker AI Ground Truth Classification Job Output. Un archivo de manifiesto está compuesto por una o más líneas JSON, una para cada imagen que desee importar.

Para crear un archivo de manifiesto para imágenes clasificadas

1. Crea un archivo de texto vacío.
2. Agregue una línea JSON para cada imagen que quiera importar. La línea JSON debería tener un aspecto similar al siguiente:

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

3. Guarde el archivo.

> **Note**
>
> 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 Creación de un archivo de manifiesto.

Clasificación: líneas JSON

En esta sección, aprenderá a crear una línea JSON que clasifique una imagen como normal o anómala.

Importación de la detección de anomalías

La siguiente línea JSON muestra una imagen etiquetada como anomalía. Tenga en cuenta que el valor de `class-name` es `anomaly`.

```
{
    "source-ref": "s3: //bucket/image/anomaly/abnormal-1.jpg",
    "anomaly-label-metadata": {
        "confidence": 1,
```

```
        "job-name": "labeling-job/auto-label",
        "class-name": "anomaly",
        "human-annotated": "yes",
        "creation-date": "2020-11-10T03:37:09.600",
        "type": "groundtruth/image-classification"
    },
    "anomaly-label": 1
}
```

Importe

La siguiente línea JSON muestra una imagen etiquetada como normal. Tenga en cuenta que el valor de `class-name` es `normal`.

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

Claves y valores de línea JSON

La siguiente información describe las claves y los valores de una línea JSON de Amazon Lookout for Vision.

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.

Uso de la detección de anomalías

(Obligatorio) El atributo de etiqueta. Utilice la clave `anomaly-label` u otro nombre de clave que elija. Amazon Lookout for Vision requiere el valor clave (`0`en el ejemplo anterior), pero no lo utiliza.

El manifiesto de salida creado por Amazon Lookout for Vision convierte el valor en 1 para una imagen anómala y en un valor `0` de para una imagen normal. El valor de `class-name` determina si la imagen es normal o anómala.

Debe haber los metadatos correspondientes identificados por el nombre del campo con -metadata anexado. Por ejemplo, `"anomaly-label-metadata"`.

anomaly-label-metadata

(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

(Opcional) Amazon Lookout for Vision no lo usa actualmente Si especifica un valor, utilice un valor de 1.

job-name

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

class-name

(Obligatorio) Si la imagen contiene contenido normal, especifique `normal` o, de lo contrario, especifique `anomaly`. Si el valor de `class-name` es cualquier otro valor, la imagen se añade al conjunto de datos como una imagen sin etiqueta. Para etiquetar una imagen, consulte Añadir imágenes a su conjunto de datos..

anotado por humanos

(Obligatorio) Especifique `"yes"` si la anotación la completó un humano. De lo contrario, especifique `"no"`.

Fecha de creación

(Opcional) La fecha y la hora de tiempo universal coordinado (UTC) en que se creó la etiqueta.

type

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

Definir líneas JSON para la segmentación de imágenes

Defina una línea JSON para cada imagen que desee utilizar en un archivo de manifiesto de Amazon Lookout for Vision. Si desea crear un modelo de segmentación, la línea JSON debe incluir información de segmentación y clasificación de la imagen. Un archivo de manifiesto está compuesto por una o más líneas JSON, una para cada imagen que desee importar.

Para crear un archivo de manifiesto para imágenes segmentadas

1. Crea un archivo de texto vacío.
2. Agregue una línea JSON para cada imagen que quiera importar. La línea JSON debería tener un aspecto similar al siguiente:

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

3. Guarde el archivo.

> **Note**
>
> 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 Creación de un archivo de manifiesto.

Segmentación: líneas JSON

En esta sección, aprenderá a crear una línea JSON que incluya información de segmentación y clasificación de una imagen.

La siguiente línea JSON muestra una imagen con información de segmentación y clasificación. `anomaly-label-metadata`contiene información de clasificación. `anomaly-mask-refy anomaly-mask-ref-metadata` contienen información de segmentación.

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

Claves y valores de línea JSON

La siguiente información describe las claves y los valores de una línea JSON de Amazon Lookout for Vision.

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.

Uso de la detección de anomalías

(Obligatorio) El atributo de etiqueta. Utilice la clave `anomaly-label` u otro nombre de clave que elija. Amazon Lookout for Vision requiere el valor clave (`1`en el ejemplo anterior), pero no lo utiliza. El manifiesto de salida creado por Amazon Lookout for Vision convierte el valor en `1` para una imagen anómala y en un valor `0` de para una imagen normal. El valor de `class-name` determina si la imagen es normal o anómala.

Debe haber los metadatos correspondientes identificados por el nombre del campo con -metadata anexado. Por ejemplo, `"anomaly-label-metadata"`.

anomaly-label-metadata

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

confidence

(Opcional) Amazon Lookout for Vision no lo usa actualmente Si especifica un valor, utilice un valor de `1`.

job-name

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

class-name

(Obligatorio) Si la imagen contiene contenido normal, especifique `normal` o, de lo contrario, especifique `anomaly`. Si el valor de `class-name` es cualquier otro valor, la imagen se añade al conjunto de datos como una imagen sin etiqueta. Para etiquetar una imagen, consulte Añadir imágenes a su conjunto de datos..

anotado por humanos

(Obligatorio) Especifique `"yes"` si la anotación la completó un humano. De lo contrario, especifique `"no"`.

Fecha de creación

(Opcional) La fecha y la hora de tiempo universal coordinado (UTC) en que se creó la etiqueta.

type

(Obligatorio) El tipo de procesamiento que se debe aplicar a la imagen. Use el valor `"groundtruth/image-classification"`.

anomaly-mask-ref

(Obligatorio) La ubicación de Amazon S3 de imagen de la máscara. `anomaly-mask-ref`Utilícela para el nombre de la clave o utilice el nombre de clave que elija. La clave debe terminar con`-ref`. La imagen de la máscara debe contener máscaras de colores para cada tipo `internal-color-map` de anomalía. El formato es `"s3://`*`BUCKET`*`/`*`OBJECT_PATH`*`"`. Las imágenes de un conjunto de datos importado deben almacenarse en el mismo bucket de Amazon S3. La imagen de la máscara debe ser una imagen con formato de gráfico de red portátil (PNG).

anomaly-mask-ref-metadata

(Obligatorio) Metadatos de segmentación de la imagen. `anomaly-mask-ref-metadata`Utilícelo para el nombre de la clave o utilice el nombre de clave que elija. El nombre de la clave debe terminar por`-ref-metadata`.

internal-color-map

(Obligatorio) Un mapa de colores que se asigna a tipos de anomalías individuales. Los colores deben coincidir con los colores de la imagen de la máscara (`anomaly-mask-ref`).

clave

(Obligatorio) La clave del mapa. La entrada `0` debe contener el nombre de la clase BACKGROUND, que representa las áreas fuera de las anomalías en la imagen.

class-name

(Obligatorio) El nombre del tipo de anomalía, como un arañazo o una abolladura.

color hexadecimal

(Obligatorio) El color hexadecimal del tipo de anomalía, por ejemplo `#2ca02c`. El color debe coincidir con uno de los colores. `anomaly-mask-ref` El valor del tipo de `BACKGROUND` anomalía es siempre`#ffffff`.

confidence

(Obligatorio) Amazon Lookout for Vision no lo utiliza actualmente, pero se requiere un valor flotante.

anotado por humanos

(Obligatorio) Especifique `"yes"` si la anotación la completó un humano. De lo contrario, especifique `"no"`.

Fecha de creación

(Opcional) La fecha y la hora de tiempo universal coordinado (UTC) en que se creó la información de segmentación.

type

(Obligatorio) El tipo de procesamiento que se debe aplicar a la imagen. Use el valor `"groundtruth/semantic-segmentation"`.

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

Este script de Python de ejemplo simplifica la creación de un archivo de manifiesto de clasificación mediante el uso de un archivo de valores separados por comas (CSV) para etiquetar las imágenes. Cree el archivo CSV.

Un archivo de manifiesto describe las imágenes utilizadas para entrenar un modelo. Un archivo de manifiesto se compone de una o más líneas JSON. Cada línea JSON describe una sola imagen. Para obtener más información, consulte Definir líneas JSON para la clasificación de imágenes.

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. Para este script, cada fila del archivo CSV incluye la ubicación en S3 de una imagen y la clasificación de las anomalías de la imagen (`normaloanomaly`). Cada fila se asigna a una línea JSON del archivo de manifiesto.

Por ejemplo, el siguiente archivo CSV describe algunas de las imágenes de las imágenes de ejemplo.

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

El script genera líneas JSON para cada fila. Por ejemplo, la siguiente es la línea JSON de la primera fila (`s3://s3bucket/circuitboard/train/anomaly/train-anomaly_1.jpg,anomaly`).

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

Si el archivo CSV no incluye la ruta de Amazon S3 para las imágenes, utilice el argumento de la línea de `--s3-path` comandos para especificar la ruta de Amazon S3 a las imágenes.

Antes de crear el archivo de manifiesto, el script comprueba si hay imágenes duplicadas en el archivo CSV y cualquier clasificación de imágenes que no sea `normal` ni `anomaly`. Si se encuentran imágenes duplicadas o errores de clasificación de imágenes, el script hace lo siguiente:

- Registra la primera entrada de imagen válida para todas las imágenes de un archivo CSV desduplicado.
- Registra las apariciones duplicadas de una imagen en el archivo de errores.
- Registra las clasificaciones de imágenes que no están en el archivo de errores `normal` o que están `anomaly` incluidas en él.
- No crea un archivo de manifiesto.

El archivo de errores incluye el número de línea en el que se encuentra una imagen duplicada o un error de clasificación en el archivo CSV de entrada. Utilice el archivo CSV de errores para actualizar el archivo CSV de entrada y, a continuación, vuelva a ejecutar el script. Como alternativa, utilice el archivo CSV de errores para actualizar el archivo CSV desduplicado, que solo contiene entradas de imágenes únicas e imágenes sin errores de clasificación de imágenes. Vuelva a ejecutar el script con el archivo CSV desduplicado actualizado.

Si no se encuentran duplicados ni errores en el archivo CSV de entrada, el script elimina el archivo CSV de la imagen desduplicada y el archivo de errores, 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. El script se ha probado con la versión 3.7 de Python.

Para crear un archivo de manifiesto desde 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
El nombre de la imagen o la ruta de Amazon S3 a la imagen. Por ejemplo, `s3://s3bucket/circuitboard/train/anomaly/train-anomaly_10.jpg` . No puede haber una mezcla de imágenes con la ruta de Amazon S3 e imágenes sin ella.	La clasificación de anomalías de la imagen (`normal` o `anomaly`).

   Por ejemplo, `s3://s3bucket/circuitboard/train/anomaly/image_10.jpg,anomaly` o `image_11.jpg,normal`
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 1.
   - (Opcional)`--s3-path` *`s3://path_to_folder/`*: la ruta de Amazon S3 que se va a añadir a los nombres de los archivos de imagen (campo 1). `--s3-path`Úselo si las imágenes del campo 1 aún no contienen una ruta S3.

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

```
s3://s3bucket/circuitboard/train/anomaly/train_11.jpg,anomaly
s3://s3bucket/circuitboard/train/normal/train_1.jpg,normal

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

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

logger = logging.getLogger(__name__)


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

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

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

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

        reader = csv.reader(input_file, delimiter=',')
        dedup_writer = csv.writer(dedup)
        error_writer = csv.writer(errors)
        line = 1
        entries = set()
```

```
        for row in reader:

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

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

            # Write first image entry to dedup file and record duplicates.
            key = row[0]
            if key not in entries:
                dedup_writer.writerow(row)
                entries.add(key)
            else:
                error_writer.writerow([line, row[0], row[1], "DUPLICATE"])
                errors_found = True
            line += 1

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

    return errors_found


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

    image_count = 0
    anomalous_count = 0
```

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

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

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

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

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

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

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

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

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

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

```
    return image_count, anomalous_count


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

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

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


def main():

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

    try:

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

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

        # Create manifest file if there are no duplicate images.
        if check_errors(csv_file):
            print(f"Issues found. Use {csv_file_no_extension}_errors.csv "\
                "to view duplicates and errors.")
            print(f"{csv_file}_deduplicated.csv contains the first"\
```

```
                "occurrence of a duplicate.\n"
                  "Update as necessary with the correct information.")
            print(f"Re-run the script with
 {csv_file_no_extension}_deduplicated.csv")
        else:
            print('No duplicates found. Creating manifest file.')

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

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

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

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

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

4. Si se producen imágenes duplicadas o errores de clasificación:

   a. Utilice el archivo de errores para actualizar el archivo CSV desduplicado o el archivo CSV de entrada.

   b. Vuelva a ejecutar el script con el archivo CSV desduplicado actualizado o el archivo CSV de entrada actualizado.

5. Si planea usar un conjunto de datos de prueba, repita los pasos del 1 al 4 para crear un archivo de manifiesto para su conjunto de datos de prueba.

6. Si es necesario, copie las imágenes de su equipo a la ruta del bucket de Amazon S3 que especificó en la columna 1 del archivo CSV (o que especificó en la línea de comandos `--s3-path`). Para copiar las imágenes, ingrese el siguiente comando en el símbolo del sistema.

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

7. Siga las instrucciones que se indican en Creación de un conjunto de datos con un archivo de manifiesto (consola) para crear un conjunto de datos. Si utiliza el AWS SDK, consulteCreación de un conjunto de datos con un archivo de manifiesto (SDK).

Creación de un conjunto de datos con un archivo de manifiesto (consola)

El siguiente procedimiento muestra cómo crear un conjunto de datos de entrenamiento o prueba mediante la importación de un archivo de manifiesto en formato SageMaker AI que se almacena en un bucket de Amazon S3.

Tras crear el conjunto de datos, puede añadir más imágenes al conjunto de datos o añadir etiquetas a las imágenes. Para obtener más información, consulte Añadir imágenes a su conjunto de datos..

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

1. Crea o usa un archivo de manifiesto existente en formato AI Ground Truth SageMaker compatible con Amazon Lookout for Vision. Para obtener más información, consulte Creación de un archivo de manifiesto.
2. Inicie sesión en la consola de Amazon S3 AWS Management Console y ábrala en https://console.aws.amazon.com/s3/.
3. En un bucket de Amazon S3, cree una carpeta para guardar el archivo de manifiesto.
4. Suba el archivo de manifiesto a la carpeta que acaba de crear.
5. En un bucket de Amazon S3, cree una carpeta para guardar sus imágenes.
6. Para crear una carpeta de trabajo, puede crear una carpeta de Amazon.

   > **Important**
   >
   > El valor del campo `source-ref` de cada línea JSON debe asignarse a las imágenes de la carpeta.

7. Abre la consola https://console.aws.amazon.com/lookoutvision/Amazon Lookout for Vision en.
8. Elija Comenzar.
9. En el panel de navegación izquierdo, elija Proyectos.
10. Elija el proyecto que desea agregar para usarlo con el archivo de manifiesto.
11. En la sección Cómo funciona, seleccione Crear conjunto de datos.

12. Seleccione la pestaña Conjunto de datos único o la pestaña Separar conjuntos de datos de entrenamiento y prueba y siga los pasos.

    Single dataset

    1. Elija Crear un solo conjunto de datos.
    2. En la sección Configuración de la fuente de la imagen, selecciona Importar imágenes etiquetadas por SageMaker Ground Truth.
    3. Para la ubicación del archivo. manifest, introduzca la ubicación del archivo de manifiesto.

    Separate training and test datasets

    1. Elija Crear un conjunto de datos de entrenamiento y un conjunto de datos de prueba.
    2. En la sección de detalles del conjunto de datos de entrenamiento, selecciona Importar imágenes etiquetadas por SageMaker Ground Truth.
    3. En la ubicación del archivo. manifest, introduzca la ubicación del archivo de manifiesto de entrenamiento.
    4. En la sección Detalles del conjunto de datos de prueba, selecciona Importar imágenes etiquetadas por SageMaker Ground Truth.
    5. En la ubicación del archivo. manifest, introduzca la ubicación del archivo de manifiesto de prueba.

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

13. Elija Enviar.
14. Siga los pasos que se indican en Entrenamiento del modelo para entrenar su modelo.

Amazon Lookout for Vision crea un conjunto de datos en la carpeta de `datasets` bucket de Amazon S3. El archivo `.manifest` original permanece inalterado.

Creación de un conjunto de datos con un archivo de manifiesto (SDK)

La CreateDatasetoperación se utiliza para crear los conjuntos de datos asociados a un proyecto de Amazon Lookout for Vision.

Si desea utilizar un único conjunto de datos para el entrenamiento y las pruebas, cree un único conjunto de datos con el valor `DatasetType` establecido en `train`. Durante el entrenamiento, el conjunto de datos se divide internamente para crear un conjunto de datos de entrenamiento y prueba. No tiene acceso a los conjuntos de datos de entrenamiento y de prueba divididos. Si quiere un conjunto de datos de prueba independiente, realice una segunda llamada a `CreateDataset` con el valor `DatasetType` establecido en `test`. Durante el entrenamiento, los conjuntos de datos de entrenamiento y prueba se utilizan para entrenar y probar el modelo.

Si lo desea, puede utilizar el `DatasetSource` parámetro para especificar la ubicación de un archivo de manifiesto en formato SageMaker AI Ground Truth que se utiliza para rellenar el conjunto de datos. En este caso, la llamada a `CreateDataset` es asíncrona. Para comprobar el estado actual, llame a `DescribeDataset`. Para obtener más información, consulte Visualización de sus conjuntos de datos. Si se produce un error de validación durante la importación, el valor de `Status` se establece en CREATE_FAILED y el mensaje de estado () se establece. `StatusMessage`

> **Tip**
>
> Si va a crear un conjunto de datos con el conjunto de datos de ejemplo de introducción, utilice el archivo de manifiesto (`getting-started/dataset-files/manifests/train.manifest`) en el que se crea el script. Paso 1: Crear el archivo de manifiesto y cargar las imágenes
>
> Si va a crear un conjunto de datos con las imágenes de ejemplo de la placa de circuito, tiene dos opciones:
>
> 1. Cree el archivo de manifiesto mediante código. El cuaderno de Python de Amazon Lookout for Vision Lab muestra cómo crear el archivo de manifiesto para las imágenes de ejemplo de la placa de circuito. Como alternativa, puedes usar el código de ejemplo de los conjuntos de datos del repositorio de ejemplos AWS de código.
> 2. Si ya ha utilizado la consola de Amazon Lookout for Vision para crear un conjunto de datos con las imágenes de ejemplo de la placa de circuito, reutilice los archivos de manifiesto que Amazon Lookout for Vision creó para usted. Las ubicaciones de los archivos de manifiesto de entrenamiento y prueba son `s3://`*`bucket`*`/datasets/`*`project name`*`/`*`train or test`*`/manifests/output/output.manifest`.

Si no especifica `DatasetSource`, se crea un conjunto de datos vacío. En este caso, la llamada a `CreateDataset` es sincrónica. Más adelante, puedes etiquetar las imágenes en el conjunto de

datos llamando UpdateDatasetEntries. Para ver el código de ejemplo, consulte Cómo añadir más imágenes (SDK).

Si quieres reemplazar un conjunto de datos, primero elimina el conjunto de datos existente por uno nuevo del mismo tipo de conjunto de datos DeleteDataset y, a continuación, crea uno nuevo del mismo tipo de conjunto de datos mediante una llamada`CreateDataset`. Para obtener más información, consulte Eliminación de un conjunto de datos.

Después de crear los conjuntos de datos, puede crear el modelo. Para obtener más información, consulte Entrenamiento de un modelo (SDK).

Puedes ver las imágenes etiquetadas (líneas JSON) dentro de un conjunto de datos llamando ListDatasetEntries. Puede añadir imágenes etiquetadas llamando a `UpdateDatasetEntries`.

Para ver información sobre los conjuntos de datos de prueba y entrenamiento, consulte Visualización de sus conjuntos de datos.

Para crear 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: Configura el AWS CLI y AWS SDKs.
2. Cree un conjunto de datos con el siguiente ejemplo.

   CLI

   Cambie los siguientes valores:

   - `project-name`al nombre del proyecto al que desea asociar el conjunto de datos.
   - `dataset-type`al tipo de conjunto de datos que desea crear (`train`o`test`).
   - `dataset-source` a la ubicación del archivo de manifiesto de Amazon S3.
   - `Bucket` al nombre del bucket de Amazon S3 que contiene el archivo de manifiesto.
   - `Key`a la ruta y el nombre de archivo del archivo de manifiesto en el bucket de Amazon S3.

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

Python

Este código se ha tomado del GitHub repositorio de ejemplos del SDK de AWS documentación. Consulte el ejemplo completo aquí.

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

        :param lookoutvision_client: A Lookout for Vision Boto3 client.
        :param project_name: The name of the project in which you want to
                             create a dataset.
        :param bucket: The bucket that contains the manifest file.
        :param manifest_file: The path and name of the manifest file.
        :param dataset_type: The type of the dataset (train or test).
        """
        try:
            bucket, key = manifest_file.replace("s3://", "").split("/", 1)
            logger.info("Creating %s dataset type...", dataset_type)
            dataset = {
                "GroundTruthManifest": {"S3Object": {"Bucket": bucket, "Key":
 key}}
            }
            response = lookoutvision_client.create_dataset(
                ProjectName=project_name,
                DatasetType=dataset_type,
                DatasetSource=dataset,
            )
            logger.info("Dataset Status: %s", response["DatasetMetadata"]
["Status"])
            logger.info(
                "Dataset Status Message: %s",
                response["DatasetMetadata"]["StatusMessage"],
            )
            logger.info("Dataset Type: %s", response["DatasetMetadata"]
["DatasetType"])

            # Wait until either created or failed.
            finished = False
            status = ""
            dataset_description = {}
```

```
            while finished is False:
                dataset_description = lookoutvision_client.describe_dataset(
                    ProjectName=project_name, DatasetType=dataset_type
                )
                status = dataset_description["DatasetDescription"]["Status"]

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

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

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

Java V2

Este código se ha tomado del GitHub repositorio de ejemplos del SDK de AWS documentación. Consulte el ejemplo completo aquí.

```
/**
 * Creates an Amazon Lookout for Vision dataset from a manifest file.
 * Returns after Lookout for Vision creates the dataset.
 *
 * @param lfvClient    An Amazon Lookout for Vision client.
 * @param projectName  The name of the project in which you want to create a
 *                     dataset.
```

```
     * @param datasetType  The type of dataset that you want to create (train or
     *                     test).
     * @param bucket       The S3 bucket that contains the manifest file.
     * @param manifestFile The name and location of the manifest file within the S3
     *                     bucket.
     * @return DatasetDescription The description of the created dataset.
     */
    public static DatasetDescription createDataset(LookoutVisionClient lfvClient,
                    String projectName,
                    String datasetType,
                    String bucket,
                    String manifestFile)
                    throws LookoutVisionException, InterruptedException {

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

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

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

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

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

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

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

```
                                .projectName(projectName)
                                .datasetType(datasetType)
                                .build();
        }

        lfvClient.createDataset(createDatasetRequest);

        DatasetDescription datasetDescription = null;

        boolean finished = false;

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

                datasetDescription = describeDataset(lfvClient, projectName,
datasetType);

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

                                TimeUnit.SECONDS.sleep(5);

                                break;

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

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

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

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

                }
        }

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

        return datasetDescription;

}
```

3. Entrene su modelo siguiendo los pasos que se indican en Entrenamiento de un modelo (SDK).

Etiquetado de imágenes

Puede utilizar la consola de Amazon Lookout for Vision para añadir o modificar las etiquetas asignadas a las imágenes de su conjunto de datos. Si utilizas el SDK, las etiquetas forman parte del archivo de manifiesto que envíes CreateDataset. Puedes actualizar las etiquetas de una imagen llamando UpdateDatasetEntries. Para ver el código de ejemplo, consulte Cómo añadir más imágenes (SDK).

Elegir el tipo de modelo

Las etiquetas que asignas a las imágenes determinan el tipo de modelo que crea Lookout for Vision. Si su proyecto tiene un conjunto de datos de prueba independiente, etiquete las imágenes de la misma manera.

Ajuste de un modelo de clasificación de imágenes

Para crear un modelo de clasificación de imágenes, clasifique cada imagen como normal o anómala. Para cada imagen, haga Clasificación de imágenes (consola).

Modelo de segmentación de imágenes

Para crear un modelo de segmentación de imágenes, clasifique cada imagen como normal o anómala. Para cada imagen anómala, también debe especificar una máscara de píxeles para cada área anómala de la imagen y una etiqueta de anomalía para el tipo de anomalía dentro de la máscara de píxeles. Por ejemplo, la máscara azul de la siguiente imagen indica la ubicación de un tipo de anomalía por arañazo en un automóvil. Puede especificar más de un tipo de etiqueta de anomalía en una imagen. Para cada imagen, haga Segmentación de imágenes (consola).

Clasificación de imágenes (consola)

La consola de Lookout for Vision se utiliza para clasificar las imágenes de un conjunto de datos como normales o anómalas. Las imágenes no clasificadas no se utilizan para entrenar el modelo.

Si va a crear un modelo de segmentación de imágenes, omita este procedimiento y haga Segmentación de imágenes (consola), que incluye los pasos para clasificar las imágenes.

> **Note**
>
> Si acaba de completar Creación de un conjunto de datos, la consola debería mostrar el panel de control del modelo y no es necesario que realice los pasos del 1 al 4.

Para clasificar las imágenes (consola)

1. Abre la consola https://console.aws.amazon.com/lookoutvision/Amazon Lookout for Vision en.

2. En el panel de navegación izquierdo, elija Proyectos.
3. En la página Proyectos, elija el proyecto que desee usar.
4. En el panel de navegación izquierdo del proyecto, elija Conjunto de datos.
5. Si tiene conjuntos de datos de entrenamiento y de prueba independientes, elija la pestaña del conjunto de datos que quiera usar.
6. Seleccione Empezar a etiquetar.
7. Elija Seleccionar todas las imágenes de esta página.
8. Si las imágenes son normales, seleccione Clasificar como normal; de lo contrario, elija Clasificar como anomalía. Aparece una etiqueta debajo de cada imagen.
9. Si necesita cambiar la etiqueta de una imagen, siga este procedimiento:

    a. Seleccione Anomalía o Normal debajo de la imagen.

    b. Si no puede determinar la etiqueta correcta para una imagen, amplíe la imagen seleccionándola en la galería.

> **Note**
>
> Puede filtrar las etiquetas de las imágenes seleccionando la etiqueta deseada, o el estado de la etiqueta, en la sección Filtros.

10. Repita los pasos del 7 al 9 en cada página según sea necesario hasta que todas las imágenes del conjunto de datos se hayan etiquetado correctamente.
11. Elija Guardar cambios.
12. Si ha terminado de etiquetar sus imágenes, puede entrenar su modelo.

Segmentación de imágenes (consola)

Si va a crear un modelo de segmentación de imágenes, debe clasificar las imágenes como normales o anómalas. También debe añadir información de segmentación a las imágenes anómalas. Para especificar la información de segmentación, primero debe especificar las etiquetas de anomalía para cada tipo de anomalía, como una abolladura o un arañazo, que desee que busque el modelo. A continuación, especifique una máscara de anomalía y una etiqueta de anomalía para cada anomalía en las imágenes anómalas del conjunto de datos.

> **Note**
>
> Si va a crear un modelo de clasificación de imágenes, no necesita segmentar imágenes ni especificar etiquetas de anomalías.

Temas

Importe de la detección de anomalías

Defina una etiqueta de anomalía para cada tipo de anomalía que haya en las imágenes del conjunto de datos.

Describir de la detección de anomalías

1. Abre la consola https://console.aws.amazon.com/lookoutvision/Amazon Lookout for Vision en.
2. En el panel de navegación izquierdo, elija Proyectos.
3. En la página Proyectos, elija el proyecto que desee usar.
4. En el panel de navegación izquierdo del proyecto, elija Conjunto de datos.
5. En Etiquetas de anomalías, seleccione Agregar etiquetas de anomalías. Si ya ha añadido una etiqueta de anomalía, seleccione Administrar.
6. En el cuadro de diálogo de , haga lo siguiente:
    a. Introduzca la etiqueta de anomalía que desee añadir y seleccione Agregar etiqueta de anomalía.
    b. Repita el paso anterior hasta que haya introducido todas las etiquetas de anomalía que desee que busque el modelo.
    c. (Opcional) Seleccione el icono de edición para cambiar el nombre de la etiqueta.
    d. (Opcional) Seleccione el icono de eliminación para eliminar una nueva etiqueta de anomalía. No puede eliminar los tipos de anomalías que su conjunto de datos utiliza actualmente.
7. Seleccione Confirmar para añadir las nuevas etiquetas de anomalías al conjunto de datos.

Después de especificar las etiquetas de las anomalías, etiquete las imágenes de la siguiente manera. Importación de una tabla

Importación de una tabla

Para etiquetar una imagen para segmentarla, clasifique la imagen como normal o como una anomalía. A continuación, utilice la herramienta de anotación para segmentar la imagen dibujando máscaras que cubran bien las áreas de cada tipo de anomalía presente en la imagen.

Para etiquetar una imagen

1. Si tiene conjuntos de datos de entrenamiento y de prueba independientes, elija la pestaña del conjunto de datos que quiera usar.
2. Si aún no lo ha hecho, especifique los tipos de anomalías de su conjunto de datos de la siguiente manera. Importe de la detección de anomalías
3. Seleccione Empezar a etiquetar.
4. Elija Seleccionar todas las imágenes de esta página.
5. Si las imágenes son normales, seleccione Clasificar como normal; de lo contrario, elija Clasificar como anomalía.
6. Para cambiar la etiqueta de una sola imagen, seleccione Normal o Anomalía debajo de la imagen.

   > **Note**
   >
   > Puede filtrar las etiquetas de las imágenes seleccionando la etiqueta deseada, o el estado de la etiqueta, en la sección Filtros. Puede ordenar por puntuación de confianza en la sección de opciones de clasificación.

7. Para cada imagen anómala, elija la imagen para abrir la herramienta de anotación. Añada información de segmentación haciendo lo siguiente. Segmentar una imagen con la herramienta de anotación
8. Seleccione Save changes (Guardar cambios).
9. Si ha terminado de etiquetar sus imágenes, puede entrenar su modelo.

Segmentar una imagen con la herramienta de anotación

La herramienta de anotación se utiliza para segmentar una imagen marcando las áreas anómalas con una máscara.

Para segmentar una imagen con la herramienta de anotación

1. Abra la herramienta de anotación seleccionando la imagen en la galería de conjuntos de datos. Si es necesario, elija Comenzar a etiquetar para entrar en el modo de etiquetado.
2. En la sección Etiquetas de anomalías, elija la etiqueta de anomalía que desee marcar. Si es necesario, seleccione Agregar etiquetas de anomalías para añadir una nueva etiqueta de anomalía.
3. Elija una herramienta de dibujo en la parte inferior de la página y dibuje máscaras que cubran herméticamente las áreas anómalas para la etiqueta de anomalía. La siguiente imagen es un ejemplo de una máscara que cubre bien una anomalía.

A continuación, se muestra un ejemplo de una máscara de mala calidad que no cubre bien una anomalía.

4. Si tiene más imágenes que segmentar, elija Siguiente y repita los pasos 2 y 3.
5. Seleccione Enviar y cerrar para terminar de segmentar las imágenes.

Entrenamiento del modelo

Después de crear los conjuntos de datos y etiquetar las imágenes, puede entrenar el modelo. Como parte del proceso de entrenamiento, se utiliza un conjunto de datos de prueba. Si tiene un proyecto de conjunto de datos único, las imágenes del conjunto de datos se dividen automáticamente en un conjunto de datos de prueba y un conjunto de datos de entrenamiento como parte del proceso de entrenamiento. Si su proyecto tiene un conjunto de datos de entrenamiento y uno de prueba, se utilizan para entrenar y probar el conjunto de datos por separado.

Una vez finalizado el entrenamiento, puede evaluar el rendimiento del modelo y realizar las mejoras necesarias. Para obtener más información, consulte Mejorar su modelo Amazon Lookout for Vision.

Para entrenar el modelo, Amazon Lookout for Vision hace una copia de las imágenes de origen de entrenamiento y prueba. De forma predeterminada, las imágenes copiadas se cifran con una clave que AWS posee y administra. También puede optar por utilizar su propia clave de AWS Key Management Service (KMS). Para obtener más información, consulte Conceptos de AWS Key Management Service. Las imágenes de origen no se ven afectadas.

Puede asignar metadatos a su modelo en forma de etiquetas. Para obtener más información, consulte Etiquetado de modelos.

Cada vez que entrena un modelo, se crea una nueva versión del modelo. Si ya no necesita una versión de un modelo, puede eliminarla. Para obtener más información, consulte Eliminación de un modelo.

Se le cobrará por el tiempo que tarda en entrenarse correctamente el modelo. Para obtener más información, consulte Horas de entrenamiento.

Para ver los modelos existentes en un proyecto, Visualización de los modelos.

> **Note**
>
> Si acaba de completar Creación de un conjunto de datos oAñadir imágenes a su conjunto de datos.. En este momento, la consola debería mostrar el panel de control del modelo y no es necesario que realices los pasos del 1 al 4.

Temas

- Entrenamiento de un modelo (consola)
- Entrenamiento de un modelo (SDK)

Entrenamiento de un modelo (consola)

En el siguiente procedimiento se muestra cómo verificar un dominio utilizando la consola.

Para entrenar su modelo (consola)

1. Abre la consola https://console.aws.amazon.com/lookoutvision/Amazon Lookout for Vision en.
2. En el panel de navegación izquierdo, elija Proyectos.
3. En la página Proyectos, elija el proyecto que contiene el modelo entrenado que desee entrenar.
4. En la página de detalles del proyecto, elija Entrenar modelo. El botón Entrenar modelo está disponible si tiene suficientes imágenes etiquetadas para entrenar el modelo. Si el botón no está disponible, añada más imágenes hasta que tenga suficientes imágenes etiquetadas.
5. (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.

6. (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.

7. Elija Entrenar modelo.
8. En el cuadro de diálogo ¿Quiere entrenar su modelo?, escoja Entrenar modelo.
9. En la vista de modelos, puede ver que el entrenamiento ha comenzado y comprobar el estado actual consultando la columna `Status` correspondiente a la versión del modelo. El entrenamiento de un modelo tarda un tiempo en completarse.
10. Cuando finalice el entrenamiento, podrá evaluar su rendimiento. Para obtener más información, consulte Mejorar su modelo Amazon Lookout for Vision.

Entrenamiento de un modelo (SDK)

La CreateModeloperación se utiliza para iniciar el entrenamiento, las pruebas y la evaluación de un modelo. Amazon Lookout for Vision entrena el modelo mediante el conjunto de datos de entrenamiento y pruebas asociado al proyecto. Para obtener más información, consulte Creación de un proyecto (SDK).

Cada vez que llame a `CreateModel`, crea una nueva versión del modelo. La respuesta de `CreateModel` incluye la versión del modelo.

Se le cobrará por cada entrenamiento de modelo exitoso. Utilice el parámetro `ClientToken` de entrada para evitar que sus usuarios repitan innecesariamente o accidentalmente el entrenamiento con el modelo. `ClientToken`es un parámetro de entrada idempotente que garantiza que `CreateModel` solo se complete una vez para un conjunto específico de parámetros. Si se repite

una llamada a `CreateModel` con el mismo `ClientToken` valor, se garantiza que el entrenamiento no se repita. Si no proporciona un valor para `ClientToken`, el AWS SDK que está utilizando lo insertará automáticamente. Esto evita que los reintentos tras un error de red inicien varios trabajos de entrenamiento, pero tendrá que proporcionar su propio valor para sus propios casos de uso. Para obtener más información, consulte CreateModel.

El entrenamiento tarda un tiempo en completarse. Para comprobar el estado actual, llame `DescribeModel` y pase el nombre del proyecto (especificado en la llamada a`CreateProject`) y la versión del modelo. El `status` campo indica el estado actual del entrenamiento del modelo. Para ver el código de ejemplo, consulte Visualización de sus modelos (SDK).

Si el entrenamiento es exitoso, puede evaluar el modelo. Para obtener más información, consulte Mejorar su modelo Amazon Lookout for Vision.

Para ver los modelos que ha creado en un proyecto, llame`ListModels`. Para ver el código de ejemplo, consulte Visualización de los modelos.

Para 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: Configura el AWS CLI y AWS SDKs.
2. Utilice el siguiente código de ejemplo para entrenar un modelo.

   CLI

   Cambie los siguientes valores:

   - `project-name`al nombre del proyecto que contiene el modelo que desea crear.
   - `output-config`a la ubicación en la que desee guardar los resultados del entrenamiento. Reemplace los siguientes valores:
     - `output bucket`con el nombre del bucket de Amazon S3 en el que Amazon Lookout for Vision guarda los resultados del entrenamiento.
     - `output folder`con el nombre de la carpeta en la que desea guardar los resultados del entrenamiento.
     - `Key` con el nombre de clave de una etiqueta.
     - `Value`con un valor al que asociarse`tag_key`.

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

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

Python

Este código se ha tomado del GitHub repositorio de ejemplos del SDK de AWS documentación. Consulte el ejemplo completo aquí.

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

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

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

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

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

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

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

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

Java V2

Este código se ha tomado del GitHub repositorio de ejemplos del SDK de AWS documentación. Consulte el ejemplo completo aquí.

```
/**
 * Creates an Amazon Lookout for Vision model. The function returns after model
 * training completes. Model training can take multiple hours to complete.
 * You are charged for the amount of time it takes to successfully train a
 model.
 * Returns after Lookout for Vision creates the dataset.
 *
 * @param lfvClient   An Amazon Lookout for Vision client.
 * @param projectName The name of the project in which you want to create a
 *                    model.
 * @param description A description for the model.
 * @param bucket      The S3 bucket in which Lookout for Vision stores the
 *                    training results.
 * @param folder      The location of the training results within the S3
 *                    bucket.
 * @return ModelDescription The description of the created model.
 */
public static ModelDescription createModel(LookoutVisionClient lfvClient, String
 projectName,
                String description, String bucket, String folder)
                throws LookoutVisionException, InterruptedException {

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

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

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

        CreateModelRequest createModelRequest = CreateModelRequest.builder()
                        .projectName(projectName)
                        .description(description)
```

```
                        .outputConfig(config)
                        .build();

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

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

        // Wait until training finishes or fails.

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

                descriptionResponse =
lfvClient.describeModel(describeModelRequest);

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

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

                                break;

                        case TRAINING_FAILED:
                                logger.log(Level.SEVERE,
```

```
                                                "Model training failed for for
project {0} version {1}.",
                                                new Object[] { projectName,
modelVersion });
                                finished = true;
                                break;

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

descriptionResponse.modelDescription()

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

                }
        } while (!finished);

        return descriptionResponse.modelDescription();

}
```

3. Cuando finalice el entrenamiento, podrá evaluar su rendimiento. Para obtener más información, consulte Mejorar su modelo Amazon Lookout for Vision.

Solución de problemas en el entrenamiento de modelos

Los problemas con el archivo de manifiesto o las imágenes de entrenamiento pueden provocar un error en el entrenamiento del modelo. Antes de volver a entrenar su modelo, compruebe los siguientes posibles problemas.

Los colores de las etiquetas de anomalías no coinciden con el color de las anomalías en la imagen de la máscara

Si estás entrenando un modelo de segmentación de imágenes, el color de la etiqueta de anomalía del archivo de manifiesto debe coincidir con el color de la imagen de la máscara. La línea JSON

de una imagen del archivo de manifiesto tiene metadatos (`internal-color-map`) que indican a Amazon Lookout for Vision qué color corresponde a una etiqueta de anomalía. Por ejemplo, el color de la etiqueta de anomalía `scratch` de la siguiente línea JSON es `#2ca02c`.

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

Si los colores de la imagen de la máscara no coinciden con los valores indicadoshex-color, el entrenamiento no se realizará correctamente y tendrás que actualizar el archivo de manifiesto.

Para actualizar los valores de color de un archivo de manifiesto

1. Con un editor de texto, abra el archivo de manifiesto que utilizó para crear el conjunto de datos.
2. Para cada línea JSON (imagen), compruebe que los colores (hex-color) del internal-color-map campo coincidan con los colores de las etiquetas de anomalías de la imagen de la máscara.

   Puede obtener la ubicación de la imagen de la máscara a partir del *anomaly-mask*-ref campo. Descargue la imagen a su equipo y use el siguiente código para obtener los colores de una imagen.

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

3. Para cada imagen con una asignación de color incorrecta, actualiza el hex-color campo de la línea JSON de la imagen.
4. Guarde el archivo de manifiesto actualizado.
5. Elimine el conjunto de datos existente del proyecto.
6. Crea un nuevo conjunto de datos en el proyecto con el archivo de manifiesto actualizado.
7. Entrenamiento del modelo.

Como alternativa, para los pasos 5 y 6, puedes actualizar las imágenes individuales del conjunto de datos llamando a la UpdateDatasetEntriesoperación y proporcionando líneas JSON actualizadas para las imágenes que deseas actualizar. Para ver el código de ejemplo, consulte Cómo añadir más imágenes (SDK).

Las imágenes de máscara no están en formato PNG

Si está entrenando un modelo de segmentación de imágenes, las imágenes de la máscara deben estar en formato PNG. Si crea un conjunto de datos a partir de un archivo de manifiesto, asegúrese

de que las imágenes de máscara a las que hace referencia *anomaly-mask*`-ref` estén en formato PNG. Si las imágenes de la máscara no están en formato PNG, tendrás que convertirlas a formato PNG. No basta con cambiar el nombre de la extensión de un archivo de imagen a `.png`.

Las imágenes de máscara que crees en la consola Amazon Lookout for Vision o con SageMaker un trabajo de AI Ground Truth se crean en formato PNG. No necesita cambiar el formato de estas imágenes.

Para corregir imágenes de máscara que no estén en formato PNG en un archivo de manifiesto

1. Con un editor de texto, abra el archivo de manifiesto que utilizó para crear el conjunto de datos.
2. Para cada línea JSON (imagen), asegúrese de que la *anomaly-mask*`-ref` haga referencia a una imagen en formato PNG. Para obtener más información, consulte Creación de un archivo de manifiesto.
3. Guarde el archivo de manifiesto actualizado.
4. Elimine el conjunto de datos existente del proyecto.
5. Crea un nuevo conjunto de datos en el proyecto con el archivo de manifiesto actualizado.
6. Entrenamiento del modelo.

Las etiquetas de segmentación o clasificación son inexactas o faltan

La falta de etiquetas o la imprecisión de las etiquetas pueden provocar un error en el entrenamiento o crear un modelo con un rendimiento deficiente. Le recomendamos que etiquete todas las imágenes del conjunto de datos. Si no etiqueta todas las imágenes y el entrenamiento del modelo falla o su modelo tiene un rendimiento deficiente, agregue más imágenes.

Comprueba lo siguiente:

- Si va a crear un modelo de segmentación, las máscaras deben cubrir bien las anomalías de las imágenes del conjunto de datos. Para comprobar las máscaras de su conjunto de datos, consulte las imágenes en la galería de conjuntos de datos del proyecto. Si es necesario, vuelva a dibujar las máscaras de imagen. Para obtener más información, consulte Segmentación de imágenes (consola).

- Asegúrese de que las imágenes anómalas de las imágenes del conjunto de datos estén clasificadas. Si va a crear un modelo de segmentación de imágenes, asegúrese de que las imágenes anómalas tengan etiquetas de anomalías y máscaras de imagen.

Es importante recordar qué tipo de modelo (segmentación o clasificación) va a crear. Un modelo de clasificación no requiere máscaras de imagen en las imágenes anómalas. No añada máscaras a las imágenes del conjunto de datos destinadas a un modelo de clasificación.

Para actualizar las etiquetas que faltan

1. Abra la galería de conjuntos de datos del proyecto.
2. Filtra las imágenes sin etiquetar para ver qué imágenes no tienen etiquetas.
3. Realice una de las siguientes acciones:
   - Si va a crear un modelo de clasificación de imágenes, clasifique cada imagen sin etiquetar.
   - Si va a crear un modelo de segmentación de imágenes, clasifique y segmente cada imagen sin etiquetar.
4. Si va a crear un modelo de segmentación de imágenes, añada máscaras a cualquier imagen anómala clasificada a la que le falten máscaras.
5. Entrenamiento del modelo.

Si decide no corregir las etiquetas defectuosas o que faltan, le recomendamos que añada más imágenes etiquetadas o que elimine las imágenes afectadas del conjunto de datos. Puedes añadir más desde la consola o mediante la UpdateDatasetEntriesoperación. Para obtener más información, consulte Añadir imágenes a su conjunto de datos..

Si decide eliminar las imágenes, debe volver a crear el conjunto de datos sin las imágenes afectadas, ya que no puede eliminar una imagen de un conjunto de datos. Para obtener más información, consulte Eliminar imágenes del conjunto de datos.

Mejorar su modelo Amazon Lookout for Vision

Durante el entrenamiento, Lookout for Vision prueba su modelo con el conjunto de datos de prueba y utiliza los resultados para crear métricas de rendimiento. Puede usar las métricas de rendimiento para evaluar el rendimiento de su modelo. Si es necesario, puede tomar medidas para mejorar su conjunto de datos y, a continuación, volver a entrenar el modelo.

Si el modelo ya tiene la precisión que busca, puede empezar a utilizarlo. Para obtener más información, consulte Ejecución de un modelo Amazon Lookout for Vision formado.

Temas

- Paso 1: Evaluar el rendimiento de su modelo
- Paso 2: mejorar el modelo
- Visualización de métricas de rendimiento
- Verificación del modelo con una tarea de pruebas de detección

Paso 1: Evaluar el rendimiento de su modelo

Puede acceder a las métricas de rendimiento desde la consola y desde la DescribeModeloperación. Amazon Lookout for Vision proporciona un resumen de las métricas de rendimiento del conjunto de datos de prueba y los resultados previstos para todas las imágenes individuales. Si su modelo es un modelo de segmentación, la consola también muestra las métricas resumidas de cada etiqueta de anomalía.

Para ver las métricas de rendimiento y probar las predicciones de las imágenes en la consola, consulte Visualización de métricas de rendimiento (consola). Para obtener información sobre cómo acceder a las métricas de rendimiento y probar las predicciones de las imágenes con la operación `DescribeModel`, consulte Visualización de métricas de rendimiento (SDK).

Métricas de clasificación de imagen

Amazon Lookout for Vision proporciona las siguientes métricas resumidas para las clasificaciones que realiza un modelo durante las pruebas:

- Precisión

- Exhaustividad
- Puntuación F1

Métricas del modelo de segmentación de imágenes

Si el modelo es un modelo de segmentación de imágenes, Amazon Lookout for Vision proporciona un resumen de las métricas de clasificación de imágenes y un resumen de las métricas de rendimiento para cada etiqueta de anomalía:

- Puntuación F1
- Intersección sobre la unión (IoU) de media

Precisión

La métrica de precisión responde a la pregunta: cuando el modelo predice que una imagen contiene una anomalía, ¿con qué frecuencia es correcta esa predicción?

La precisión es una métrica útil en situaciones en las que el coste de un falso positivo es elevado. Por ejemplo, el coste de retirar una pieza de una máquina que no esté defectuosa de una máquina ensamblada.

Amazon Lookout for Vision proporciona un valor métrico de precisión resumido para todo el conjunto de datos de prueba.

La precisión es la fracción de las anomalías pronosticadas correctamente (positivos verdaderos) sobre todas las anomalías pronosticadas (positivos verdaderos y falsos). La fórmula de precisión es la siguiente.

Valor de precisión = positivos verdaderos/(positivos verdaderos + falsos positivos)

Los valores posibles de precisión oscilan entre 0 y 1. La consola de Amazon Lookout for Vision muestra la precisión como un valor porcentual (0—100).

Un valor de precisión más alto indica que una mayor parte de las anomalías pronosticadas son correctas. Por ejemplo, supongamos que el modelo predice que 100 imágenes son anómalas. Si 85 de las predicciones son correctas (los verdaderos positivos) y 15 son incorrectas (los falsos positivos), la precisión se calcula de la siguiente manera:

85 positivos verdaderos /(85 positivos verdaderos + 15 falsos positivos) = valor de precisión de 0,85

Sin embargo, si el modelo solo predice correctamente 40 imágenes de un total de 100 predicciones de anomalías, el valor de precisión resultante es inferior, de 0,40 (es decir, 40/(40 + 60) = 0,40). En este caso, el modelo hace más predicciones incorrectas que correctas. Para solucionar este problema, considere la posibilidad de realizar mejoras en el modelo. Para obtener más información, consulte Paso 2: mejorar el modelo.

Para obtener más información, consulte Precisión y exhaustividad.

Exhaustividad

La métrica de recuperación responde a la pregunta: del número total de imágenes anómalas del conjunto de datos de prueba, ¿cuántas se han predicho correctamente como anómalas?

La precisión es una métrica útil en situaciones en las que el coste de un falso positivo es elevado. Por ejemplo, cuando el coste de no retirar una pieza defectuosa es elevado. Amazon Lookout for Vision proporciona un valor métrico de precisión resumido para todo el conjunto de datos.

La Recuperación es la fracción de las imágenes de prueba anómalas que se detectaron correctamente. 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 datos. La fórmula de recuperación se calcula de la siguiente manera:

Valor de recuperación = positivos verdaderos/(positivos verdaderos + falsos negativos)

El rango de exhaustividad es de 0 a 1. La consola de Amazon Lookout for Vision muestra la precisión como un valor porcentual (0—100).

Un valor de recuperación más alto indica que se han identificado correctamente más imágenes anómalas. Por ejemplo, supongamos que el conjunto de datos de prueba contiene 100 imágenes anómalas. Si el modelo detecta correctamente 90 de las 100 imágenes anómalas, la recuperación es la siguiente:

90 positivos verdaderos /(90 positivos verdaderos + 10 falsos negativos) = valor de recuperación de 0,90

Un valor de recuperación de 0,90 indica que el modelo predice correctamente la mayoría de las imágenes anómalas del conjunto de datos de prueba. Si el modelo solo predice correctamente 20 de las imágenes anómalas, la recuperación es inferior a 0,20 (es decir, 20 / (20 + 80) = 0,20).

En este caso, debería plantearse la posibilidad de realizar mejoras en el modelo. Para obtener más información, consulte Paso 2: mejorar el modelo.

Para obtener más información, consulte Precisión y exhaustividad.

Puntuación F1

Amazon Lookout for Vision proporciona una puntuación media de rendimiento del modelo para el conjunto de datos de prueba. En concreto, el rendimiento del modelo para la clasificación de anomalías se mide mediante la métrica de puntuación F1, que es la media armónica de las puntuaciones de precisión y recuperación.

La puntuación F1 es una medida agregada que tiene en cuenta tanto la precisión como la memoria. 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. Por ejemplo, en un modelo con una precisión de 0,9 y una exhaustividad de 1,0, la puntuación F1 es de 0,947.

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.

Intersección sobre la unión (IoU) de media

El porcentaje medio de superposición entre las máscaras de anomalías de las imágenes de prueba y las máscaras de anomalías que el modelo predice para las imágenes de prueba. Amazon Lookout for Vision devuelve la IoU de media de cada etiqueta de anomalía y solo lo devuelven los modelos de segmentación de imágenes.

Un valor porcentual bajo indica que el modelo no hace coincidir con precisión las máscaras previstas para una etiqueta con las máscaras de las imágenes de prueba.

La siguiente imagen tiene una IoU baja. La máscara naranja es la predicción del modelo y no cubre bien la máscara azul que representa la máscara en una imagen de prueba.

La siguiente imagen tiene una IoU alta. La máscara azul (imagen de prueba) está bien cubierta por la máscara naranja (máscara prevista).

Comprobación de los resultados

Durante la prueba, el modelo predice la clasificación de cada imagen de prueba en el conjunto de datos de prueba. El resultado de cada predicción se compara con la etiqueta (normal o anormal) de la imagen de prueba correspondiente de la siguiente manera:

- Predecir correctamente que una imagen es anómala se considera un verdadero positivo.
- Predecir incorrectamente que una imagen es anómala se considera un falso positivo.
- Predecir correctamente que una imagen es normal se considera un verdadero negativo.
- Predecir incorrectamente que una imagen es normal se considera un falso negativo.

Si el modelo es un modelo de segmentación, también predice máscaras y etiquetas de anomalías para la ubicación de las anomalías en la imagen de prueba.

Amazon Lookout for Vision utiliza los resultados de las comparaciones para generar las métricas de rendimiento.

Paso 2: mejorar el modelo

Las métricas de rendimiento pueden mostrar que puede mejorar el modelo. Por ejemplo, si el modelo no detecta todas las anomalías en el conjunto de datos de prueba, el modelo tiene una recuperación baja (es decir, la métrica de recuperación tiene un valor bajo). Si necesita mejorar el modelo, tenga en cuenta lo siguiente:

- Compruebe que las imágenes del conjunto de datos de entrenamiento y prueba estén etiquetadas correctamente.
- Reduzca la variabilidad de las condiciones de captura de imágenes, como la iluminación y la postura del objeto, y entrene su modelo con objetos del mismo tipo.
- Asegúrese de que sus imágenes muestren solo el contenido requerido. Por ejemplo, si su proyecto detecta anomalías en las piezas de la máquina, asegúrese de que no haya otros objetos en las imágenes.
- Añada más imágenes etiquetadas al conjunto de datos de entrenamiento y prueba. Si su conjunto de datos de prueba tiene una capacidad de recuperación y precisión excelentes, pero el modelo tiene un rendimiento deficiente cuando se implementa, es posible que su conjunto de datos de prueba no sea lo suficientemente representativo y necesite ampliarlo.
- Si el resultado del conjunto de datos de prueba es deficiente en cuanto a memoria y precisión, tenga en cuenta si coinciden las anomalías y las condiciones de captura de imágenes en los conjuntos de datos de entrenamiento y de prueba. Si las imágenes de entrenamiento no son representativas de las anomalías y condiciones esperadas, pero las imágenes de las imágenes de prueba sí lo son, añada imágenes al conjunto de datos de entrenamiento con las anomalías y condiciones esperadas. Si las imágenes del conjunto de datos de prueba no están en las condiciones esperadas, pero las imágenes de entrenamiento sí, actualice el conjunto de datos de prueba.

  Para obtener más información, consulte Añadir más imágenes. Una forma alternativa de añadir imágenes etiquetadas al conjunto de datos de entrenamiento es ejecutar una tarea de pruebas de detección y verificar los resultados. A continuación, puede añadir las imágenes verificadas al conjunto de datos de entrenamiento. Para obtener más información, consulte Verificación del modelo con una tarea de pruebas de detección.
- Asegúrese de tener imágenes normales y anómalas lo suficientemente diversas en su conjunto de datos de entrenamiento y prueba. Las imágenes deben representar el tipo de imágenes normales y anómalas que encontrará el modelo. Por ejemplo, al analizar las placas de circuitos, las imágenes normales deberían representar las variaciones en la posición y la soldadura de los

componentes, como resistencias y transistores. Las imágenes anómalas deben representar los distintos tipos de anomalías que puede encontrar el sistema, como componentes extraviados o faltantes.

- Si su modelo tiene una IoU de media baja para los tipos de anomalías detectadas, compruebe las salidas de máscara del modelo de segmentación. En algunos casos de uso, como los rasgos, el modelo puede producir rasgos muy parecidos a los reales en las imágenes de prueba, pero con una baja superposición de píxeles. Por ejemplo, dos líneas paralelas que están separadas por una distancia de 1 píxel. En esos casos, la IoU de media es un indicador poco fiable para medir el éxito de una predicción.
- Si el tamaño de la imagen es pequeño o la resolución de la imagen es baja, considere la posibilidad de capturar imágenes con una resolución más alta. Las dimensiones de la imagen pueden oscilar entre 64 x 64 píxeles y 4096 píxeles x 4096 píxeles.
- Si el tamaño de la anomalía es pequeño, considere la posibilidad de dividir las imágenes en teselas independientes y utilícelas para el entrenamiento y las pruebas. Esto permite al modelo ver los defectos en una imagen con un tamaño mayor.

Una vez que haya mejorado su conjunto de datos de entrenamiento y pruebas, vuelva a entrenar y vuelva a evaluar el modelo. Para obtener más información, consulte Entrenamiento del modelo.

Si las métricas muestran que el modelo tiene un rendimiento aceptable, puede verificar su rendimiento añadiendo los resultados de una tarea de pruebas de detección al conjunto de datos de prueba. Tras volver a entrenarse, las métricas de rendimiento deberían confirmar las métricas de rendimiento del entrenamiento anterior. Para obtener más información, consulte Verificación del modelo con una tarea de pruebas de detección.

Visualización de métricas de rendimiento

Puede obtener las métricas de rendimiento desde la consola y llamando a la operacióncDescribeModel.

Temas

- Visualización de métricas de rendimiento (consola)
- Visualización de métricas de rendimiento (SDK)

Visualización de métricas de rendimiento (consola)

Una vez finalizada el entrenamiento, la consola muestra las métricas de rendimiento.

La consola de Amazon Lookout for Vision muestra las siguientes métricas de rendimiento para las clasificaciones realizadas durante las pruebas:

- Precisión
- Exhaustividad
- Puntuación F1

Si el modelo es un modelo de segmentación, la consola también muestra las siguientes métricas de rendimiento para cada etiqueta de anomalía:

- El número de imágenes de prueba en las que se encontró la etiqueta de anomalía.
- Puntuación F1
- Intersección sobre la unión (IoU) de media

La sección de resumen de los resultados de la prueba muestra el total de predicciones correctas e incorrectas de las imágenes del conjunto de datos de la prueba. También puede ver las asignaciones de etiquetas previstas y reales para las imágenes individuales del conjunto de datos de prueba.

El siguiente procedimiento muestra cómo obtener métricas de rendimiento de la vista de lista de modelos de un proyecto.

Para ver las métricas de desempeño (consola)

1. Abre la consola https://console.aws.amazon.com/lookoutvision/Amazon Lookout for Vision en.
2. Elija Comenzar.
3. En el panel de navegación izquierdo, elija Proyectos.
4. En la vista de proyectos, elija el proyecto que contiene la versión del modelo que desea ver.
5. En el panel de navegación izquierdo, en el nombre del proyecto, elija Modelos.
6. En la vista de lista de modelos, seleccione las versiones del modelo que desea ver.
7. En la página de detalles del modelo, consulte las métricas de rendimiento en la pestaña Métricas de rendimiento.

8. Tenga en cuenta lo siguiente:

   a. La sección de Métricas de rendimiento del modelo contiene las métricas generales del modelo (precisión, recuperación, puntuación de F1) para las predicciones de clasificación que el modelo realizó para las imágenes de prueba.

   b. Si el modelo es un modelo de segmentación de imágenes, la sección Rendimiento por etiqueta contiene el número de imágenes de prueba en las que se encontró la etiqueta de anomalía. También puede ver las métricas (puntuación F1, IoU de media) de cada etiqueta de anomalía.

   c. La sección Resumen de los resultados de las pruebas proporciona los resultados de cada imagen de prueba que Lookout for Vision utiliza para evaluar el modelo. Contiene lo siguiente:

      - El número total de predicciones de clasificación correctas (positivas verdaderas) e incorrectas (falsas negativas) (normales o anómalas) para todas las imágenes de prueba.
      - La predicción de clasificación de cada imagen de prueba. Si aparece Correcto debajo de una imagen, la clasificación prevista coincide con la clasificación real de la imagen. De lo contrario, el modelo no clasificó correctamente la imagen.
      - Con un modelo de segmentación de imágenes, verá las etiquetas de anomalías que el modelo asignó a la imagen y las ocultará en la imagen para que coincidan con los colores de las etiquetas de las anomalías.

Visualización de métricas de rendimiento (SDK)

Puede utilizar la DescribeModeloperación para obtener el resumen de las métricas de rendimiento (clasificación) del modelo, el manifiesto de evaluación y los resultados de la evaluación de un modelo.

Obtener el resumen de las métricas de rendimiento

Las métricas resumidas de rendimiento de las predicciones de clasificación realizadas por el modelo durante las pruebas (Precisión, Exhaustividad y Puntuación F1) se devuelven en el campo `Performance` devuelto por una llamada a `DescribeModel`.

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

```
        },
```

El campo `Performance` no incluye las métricas de rendimiento de las etiquetas de anomalías devueltas por un modelo de segmentación. Puede obtenerlos sobre el campo `EvaluationResult`. Para obtener más información, consulte Revisar el resultado de la evaluación.

Para obtener información sobre el resumen de las métricas de rendimiento, consulte Paso 1: Evaluar el rendimiento de su modelo. Para ver el código de ejemplo, consulte Visualización de sus modelos (SDK).

Uso del manifiesto de evaluación

El manifiesto de evaluación proporciona métricas de predicción de prueba para las imágenes individuales utilizadas para probar un modelo. Para cada imagen del conjunto de datos de prueba, una línea JSON contiene la información original de la prueba (verdad fundamental) y la predicción del modelo para la imagen. Amazon Lookout for Vision almacena el manifiesto de evaluación en un bucket de Amazon S3. Puede obtener la ubicación desde el campo `EvaluationManifest` en la respuesta de la operación `DescribeModel`.

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

El formato del nombre del archivo es `EvaluationManifest-`*`project name`*`.json`. Para ver el código de ejemplo, consulte Visualización de los modelos.

En el siguiente ejemplo de línea JSON, `class-name` es la verdad fundamental del contenido de la imagen. En este ejemplo, la imagen contiene una anomalía. El campo `confidence` muestra la confianza que Amazon Lookout for Vision tiene en la predicción.

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

    // Test dataset ground truth
```

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

Revisar el resultado de la evaluación

El resultado de la evaluación tiene las siguientes métricas de rendimiento agregadas (clasificación) para todo el conjunto de imágenes de prueba:

- Precisión
- Exhaustividad
- Curva ROC (no se muestra en la consola)
- Precisión media (no se muestra en la consola)
- Puntuación F1

El resultado de la evaluación también incluye el número de imágenes utilizadas para entrenar y probar el modelo.

Si el modelo es un modelo de segmentación, el resultado de la evaluación también incluye las siguientes métricas para cada etiqueta de anomalía que se encuentre en el conjunto de datos de la prueba:

- Precisión
- Exhaustividad
- Puntuación F1
- Intersección sobre la unión (IoU) de media

Amazon Lookout for Vision almacena el manifiesto de evaluación en un bucket de Amazon S3. Puede obtener la ubicación desde el campo `EvaluationResult` en la respuesta de la operación `DescribeModel`.

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

El formato del nombre del archivo es `EvaluationResult-`*`project name`*`.json`. Para ver un ejemplo, consulta Visualización de los modelos.

El siguiente esquema muestra el resultado de la evaluación.

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

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

Verificación del modelo con una tarea de pruebas de detección

Si desea verificar o mejorar la calidad del modelo, puede ejecutar una tarea de pruebas de detección. Una tarea de pruebas de detección detecta anomalías en las nuevas imágenes que proporcione.

Puede verificar los resultados de la detección y añadir las imágenes verificadas al conjunto de datos. Si tiene conjuntos de datos de entrenamiento y de prueba independientes, las imágenes verificadas se agregan al conjunto de datos de entrenamiento.

Puede verificar imágenes desde el equipo local o desde imágenes ubicadas en un bucket de Amazon S3. Si desea añadir imágenes verificadas al conjunto de datos, las imágenes ubicadas en un bucket de S3 deben estar en el mismo bucket de S3 que las imágenes de su conjunto de datos.

> **Note**
>
> Para ejecutar una tarea de pruebas de detección, asegúrese de que el bucket de S3 tenga activado el control de versiones. Para obtener más información, consulte Uso del control de versiones. El bucket de la consola se crea con el control de versiones activado.

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 clave de AWS Key Management Service (KMS). Para obtener más información, consulte Conceptos de AWS Key Management Service.

Temas

Ejecución de una tarea de pruebas de detección

Realice los siguientes pasos para ejecutar una tarea de pruebas de detección.

Ejecutar una prueba de detección (consola)

1. Abre la consola https://console.aws.amazon.com/lookoutvision/Amazon Lookout for Vision en.
2. Elija Comenzar.
3. En el panel de navegación izquierdo, elija Proyectos.
4. En la vista de proyectos, elija el proyecto que contiene la versión del modelo que desea ver.
5. En el panel de navegación izquierdo, en el nombre del proyecto, elija Detecciones de prueba.
6. En la vista de detecciones de prueba, seleccione Ejecutar detección de prueba.
7. En la página Ejecutar detección de prueba, introduzca un nombre para la tarea de pruebas de detección en Nombre de la tarea.
8. En Elegir modelo, elija la versión de ese modelo que desee utilizar.
9. Importe las imágenes según la fuente de las imágenes de la siguiente manera:
   - Si va a importar las imágenes de origen desde un bucket de Amazon S3, introduzca el URI de S3.

     > **Tip**
     >
     > Si utiliza las imágenes de ejemplo de Primeros pasos, utilice la carpeta extra_images. El URI de Amazon S3 es `s3://your bucket/circuitboard/extra_images`.

   - Si está cargando imágenes desde su equipo, agréguelas después de seleccionar Detectar anomalías.
10. (Opcional) Si quiere usar su propia clave de cifrado de AWS KMS, haga lo siguiente:
    a. En Cifrado de datos de imagen, elija Personalizar 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.

11. Elija Detectar anomalías y, a continuación, elija Ejecutar detección de prueba para iniciar la tarea de pruebas de detección.
12. Compruebe el estado actual en la vista de pruebas de detección. Las pruebas de detección pueden tardar un tiempo en completarse.

Verificación de los resultados de las pruebas de detección

La verificación de los resultados de una prueba de detección puede ayudarle a mejorar su modelo.

Si las métricas de rendimiento son deficientes, mejore su modelo realizando una prueba de detección y, a continuación, añada imágenes verificadas al conjunto de datos (conjunto de datos de entrenamiento, si tiene conjuntos de datos independientes).

Si las métricas de rendimiento del modelo son buenas, pero los resultados de una prueba de detección son deficientes, puede mejorar su modelo añadiendo imágenes verificadas al conjunto de datos (conjunto de datos de entrenamiento). Si tiene un conjunto de datos de prueba independiente, considere la posibilidad de añadir más imágenes al conjunto de datos de prueba.

Después de añadir imágenes verificadas al conjunto de datos, vuelva a entrenar y reevaluar el modelo. Para obtener más información, consulte Entrenamiento del modelo.

Verificar los resultados de una prueba de detección

1. Abre la consola https://console.aws.amazon.com/lookoutvision/Amazon Lookout for Vision en.
2. En el panel de navegación izquierdo, elija Proyectos.
3. En la página Proyectos, elija el proyecto que desee usar. Aparece el panel de control de su proyecto.
4. En el panel de navegación izquierdo, elija Detecciones de prueba.
5. Elija la prueba de detección que desee comprobar.
6. En la página de pruebas de detección, elija Verificar las predicciones de la máquina.
7. Elija Seleccionar todas las imágenes de esta página.

8. Si las predicciones son correctas, seleccione Verificar como correctas. De lo contrario, seleccione Verificar como incorrecto. La predicción y la puntuación de confianza de la predicción se muestran debajo de cada imagen.
9. Si necesita cambiar la etiqueta de una imagen, siga este procedimiento:
   a. Seleccione Correcto o Incorrecto debajo de la imagen.
   b. Si no puede determinar la etiqueta correcta para una imagen, amplíe la imagen seleccionándola en la galería.

> **Note**
>
> Puede filtrar las etiquetas de las imágenes seleccionando la etiqueta deseada, o el estado de la etiqueta, en la sección Filtros. Puede ordenar por puntuación de confianza en la sección de opciones de clasificación.

10. Si su modelo es un modelo de segmentación y la máscara o etiqueta de anomalía de una imagen es incorrecta, elija Área anómala debajo de la imagen y abra la herramienta de anotación. Actualice la información de segmentación de la siguiente manera. Corregir las etiquetas de segmentación con la herramienta de anotación
11. Repita los pasos del 7 al 10 en cada página según sea necesario hasta que se hayan verificado todas las imágenes.
12. Seleccione Añadir imágenes verificadas al conjunto de datos. Si tiene conjuntos de datos independientes, las imágenes se añaden al conjunto de datos de entrenamiento.
13. Reformación de su modelo. Para obtener más información, consulte the section called “Entrenamiento del modelo”.

Corregir las etiquetas de segmentación con la herramienta de anotación

La herramienta de anotación se utiliza para segmentar una imagen marcando las áreas anómalas con una máscara.

Corregir las etiquetas de segmentación de una imagen con la herramienta de anotación

1. Abra la herramienta de anotación seleccionando un área anómala debajo de una imagen en la galería de conjunto de datos.

2. Si la etiqueta de anomalía de una máscara no es correcta, selecciónela y, a continuación, elija la etiqueta de anomalía correcta en Etiquetas de anomalías. Si es necesario, seleccione Agregar etiqueta de anomalía para añadir una nueva etiqueta de anomalía.
3. Si la máscara no es correcta, elija una herramienta de dibujo en la parte inferior de la página y dibuja máscaras que cubran bien las áreas anómalas para la etiqueta de anomalía. La siguiente imagen es un ejemplo de una máscara que cubre bien una anomalía.

A continuación, se muestra un ejemplo de una máscara de mala calidad que no cubre bien una anomalía.

4. Si tiene más imágenes que corregir, elija Siguiente y repita los pasos 2 y 3.
5. Seleccione Enviar y cerrar para terminar de actualizar las imágenes.

Ejecución de un modelo Amazon Lookout for Vision formado

Para detectar anomalías en las imágenes del modelo, primero debe iniciar el modelo con la StartModeloperación. La consola Amazon Lookout for Vision AWS CLI proporciona comandos que puede utilizar para iniciar y detener el modelo. En esta sección se incluye un ejemplo de código que puede utilizar.

Una vez iniciado el modelo, puede utilizar la `DetectAnomalies` operación para detectar anomalías en una imagen. Para obtener más información, consulte Detección de anomalías en una imagen.

Temas

- Unidades de inferencia
- Zonas de disponibilidad
- Inicio del modelo Amazon Lookout for Vision
- Paro del modelo Amazon Lookout for Vision

Unidades de inferencia

Al iniciar el modelo, Amazon Lookout for Vision aprovisiona un mínimo de un recurso de cómputo, conocido como unidad de inferencia. Usted especifica el número de unidades de inferencia que se van a utilizar en el parámetro `MinInferenceUnits` de entrada a la API `StartModel`. La asignación predeterminada para un modelo es 1 unidad de inferencia.

> **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 Precios de Amazon Lookout for Vision. Si no detiene la llamada de su modelo de forma explícita StopModel, se le cobrará incluso si no analiza activamente las imágenes con su modelo.

Las transacciones por segundo (TPS) que engloba una sola unidad de inferencia se ven afectadas por lo siguiente:

- El algoritmo que Lookout for Vision usa para entrenar el modelo. Cuando entrena un modelo, se entrenan varios modelos. Lookout for Vision selecciona el modelo con el mejor rendimiento en función del tamaño del conjunto de datos y su composición de imágenes normales y anómalas.
- Las imágenes con mayor resolución exigen más tiempo de análisis.
- Las imágenes de menor tamaño (medidas en pulgadas MBs) se analizan más rápido que las imágenes más grandes.

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 Lookout for Vision ahora puede escalar el número de unidades de inferencia que utiliza el modelo. A medida que aumenta la demanda, Etiquetas personalizadas de Amazon Lookout for Vision añade unidades de inferencia adicionales al modelo y las elimina cuando la demanda decrece.

Para permitir que Etiquetas personalizadas de Lookout for Vision 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, Lookout for Vision

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 establecer el número máximo de unidades de inferencia con la consola de Lookout for Vision. En su lugar, indique el parámetro de entrada `MaxInferenceUnits` en la operación `StartModel`.

Lookout for Vision 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 a las que Lookout for Vision se escala o reduce verticalm ente.
`InServiceInferenceUnits`	El número de unidades de inferencia que utiliza el modelo.

Si `DesiredInferenceUnits` =`InServiceInferenceUnits`, Lookout for Vision no está escalando actualmente el número de unidades de inferencia.

Si `DesiredInferenceUnits` > `InServiceInferenceUnits`, Lookout for Vision se escala verticalmente hasta alcanzar el valor de `DesiredInferenceUnits`.

Si `DesiredInferenceUnits` < `InServiceInferenceUnits`, Lookout for Vision se reduce verticalmente al valor de `DesiredInferenceUnits`.

Para obtener más información sobre las métricas devueltas por Lookout for Vision y las dimensiones de filtrado, consulta Monitoring Lookout for Vision with Amazon. CloudWatch

Para saber el número máximo de unidades de inferencia que ha solicitado para un modelo, llame a DescribeModel y revise el campo `MaxInferenceUnits` en la respuesta.

Zonas de disponibilidad

Amazon Lookout for Vision 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 DetectAnomaliesse redistribuyen entre las unidades de inferencia restantes. Estas llamadas se realizan correctamente si no superan las transacciones por segundo (TPS) permitidas de las unidades de inferencia restantes. Tras AWS reparar la zona de disponibilidad, se reinician las unidades de inferencia y se restablece toda su capacidad.

Si una sola unidad de inferencia falla, Amazon Lookout for Vision 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 del modelo Amazon Lookout for Vision

Antes de poder utilizar un modelo de Amazon Lookout for Vision para detectar anomalías, primero debe iniciar el modelo. Para iniciar un modelo, debe llamar a la StartModelAPI y pasar lo siguiente:

- ProjectName— El nombre del proyecto que contiene el modelo que se quiere iniciar.
- ModelVersion— La versión del modelo que desea iniciar.
- MinInferenceUnits— El número mínimo de unidades de inferencia. Para obtener más información, consulte Unidades de inferencia.
- (Opcional) MaxInferenceUnits: el número máximo de unidades de inferencia que Amazon Lookout for Vision puede utilizar para escalar automáticamente el modelo. Para obtener más información, consulte Escalado automático de unidades de inferencia.

La consola de Amazon Lookout for Vision proporciona ejemplos de código que puede usar para iniciar y detener el modelo.

> **Note**
>
> Se le cobrará por la cantidad de tiempo de ejecución del modelo. Para detener un modelo en ejecución, consulte Paro del modelo Amazon Lookout for Vision.
> Puedes usar el AWS SDK para ver los modelos en ejecución en todas AWS las regiones en las que Lookout for Vision está disponible. Para ver un código de ejemplo, consulte find_running_models.py.

Temas

Iniciar el modelo (consola)

La consola Amazon Lookout for Vision proporciona AWS CLI un comando que puede utilizar para iniciar un modelo. Una vez iniciado el modelo, puede empezar a detectar anomalías en las imágenes. Para obtener más información, consulte Detección de anomalías en una imagen.

Cómo iniciar un modelo (consola)

1. Si aún no lo ha hecho, instale y configure el AWS CLI y el AWS SDKs. Para obtener más información, consulte Paso 4: Configura el AWS CLI y AWS SDKs.
2. Abre la consola https://console.aws.amazon.com/lookoutvision/Amazon Lookout for Vision en.
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. En la página de detalles del modelo, seleccione Usar modelo y a continuación, seleccione Integrar la API en la nube.

> **Tip**
>
> Si desea implementar su modelo en un dispositivo de borde, elija Crear un trabajo de empaquetado de modelos. Para obtener más información, consulte Empaquetado del modelo Amazon Lookout for Vision.

8. En los comandos de la CLI de AWS, copie el comando de la AWS CLI que llama`start-model`.
9. En el símbolo del sistema, escriba el comando `start-model` copiado en el paso anterior. Si va a usar el perfil `lookoutvision` para obtener las credenciales, añada el parámetro `--profile lookoutvision-access`.
10. En la consola, elija Modelos en la página de navegación izquierda.
11. Compruebe el estado actual del modelo en la columna Estado. Si el estado es Alojado, puede utilizar el modelo para detectar anomalías en las imágenes. Para obtener más información, consulte Detección de anomalías en una imagen.

Inicio del modelo Amazon Lookout for Vision (SDK)

Para iniciar un modelo, llame a la StartModeloperación.

Es posible que un modelo tarde un tiempo en iniciarse. Puede comprobar el estado actual llamando a DescribeModel. Para obtener más información, consulte Visualización de los modelos.

Para iniciar el 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: Configura el AWS CLI y AWS SDKs.
2. Utilice el código de ejemplo para iniciar un modelo.

   CLI

   Cambie los siguientes valores:

   - `project-name` al nombre del proyecto que incluye el modelo que desea iniciar.
   - `model-version` a la versión del modelo que quiere iniciar.
   - `--min-inference-units` al número de unidades de inferencia que desea utilizar.

- (Opcional) `--max-inference-units` al número máximo de unidades de inferencia que Etiquetas personalizadas de Amazon Lookout for Vision puede utilizar para escalar automáticamente el modelo.

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

Python

Este código se ha tomado del GitHub repositorio de ejemplos del SDK de AWS documentación. Consulte el ejemplo completo aquí.

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

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

            if max_inference_units is None:
                lookoutvision_client.start_model(
```

```
                ProjectName = project_name,
                ModelVersion = model_version,
                MinInferenceUnits = min_inference_units)

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

        print("Starting hosting...")

        status = ""
        finished = False

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

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

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

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

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

Java V2

Este código se ha tomado del GitHub repositorio de ejemplos del SDK de AWS documentación. Consulte el ejemplo completo aquí.

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

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

    StartModelRequest startModelRequest = null;

    if (maxInferenceUnits == null) {

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

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

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

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

   ModelDescription modelDescription = null;

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

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

       switch (modelDescription.status()) {

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

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

           TimeUnit.SECONDS.sleep(60);

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

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

```
                    new Object[] { projectName, modelVersion,
 modelDescription.status() });
            finished = true;
            break;

        }

    } while (!finished);

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

    return modelDescription;

}
```

3. Si el resultado del código es `Model is hosted and ready for use`, puede usar el modelo para detectar anomalías en las imágenes. Para obtener más información, consulte Detección de anomalías en una imagen.

Paro del modelo Amazon Lookout for Vision

Para detener un modelo en ejecución, llame a la operación `StopModel` y ejecute lo siguiente:

- Project: el nombre del proyecto que incluye el modelo que quiere detener.
- ModelVersion— La versión del modelo que quieres detener.

La consola de Amazon Lookout for Vision proporciona ejemplos de código que puede usar para detener el modelo.

> **Note**
>
> Se le cobrará por la cantidad de tiempo de ejecución del modelo.

Temas

- Detención de un modelo (consola)

- Detención del modelo Amazon Lookout for Vision (SDK)

Detención de un modelo (consola)

Realice los pasos que se indican en el procedimiento siguiente para detener el modelo usando la consola.

Detener el modelo (consola)

1. Si aún no lo ha hecho, instale y configure el AWS CLI y el AWS SDKs. Para obtener más información, consulte Paso 4: Configura el AWS CLI y AWS SDKs.
2. Abre la consola https://console.aws.amazon.com/lookoutvision/Amazon Lookout for Vision en.
3. Elija Comenzar.
4. En el panel de navegación izquierdo, elija Proyectos.
5. En la página Proyectos, elija el proyecto que contiene el modelo activo que quiera detener.
6. En la sección Modelos, elija el modelo que desee detener.
7. En la página de detalles del modelo, seleccione Usar modelo y a continuación, seleccione Integrar la API en la nube.
8. En los comandos de la CLI de AWS, copie el comando de la AWS CLI que llama`stop-model`.
9. En el símbolo del sistema, escriba el comando `stop-model` copiado en el paso anterior. Si va a usar el perfil `lookoutvision` para obtener las credenciales, añada el parámetro `--profile lookoutvision-access`.
10. En la consola, elija Modelos en la página de navegación izquierda.
11. Compruebe el Estado actual del modelo en la columna Estado. El modelo se ha detenido cuando el valor de la columna Estado es Entrenamiento finalizado.

Detención del modelo Amazon Lookout for Vision (SDK)

Para detener un modelo, llame a la StopModeloperación.

Es posible que el modelo tarde un tiempo en detenerse. Para comprobar el estado actual, use `DescribeModel`.

Para detener el 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: Configura el AWS CLI y AWS SDKs.
2. Utilice el código de ejemplo para iniciar un modelo.

   CLI

   Cambie los siguientes valores:

   - `project-name` al nombre del proyecto que incluye el modelo que desea detener.
   - `model-version` a la versión del modelo que quiere detener.

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

   Python

   Este código se ha tomado del GitHub repositorio de ejemplos del SDK de AWS documentación. Consulte el ejemplo completo aquí.

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

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

```
            status = response["Status"]
            finished = False

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

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

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

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

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

Java V2

Este código se ha tomado del GitHub repositorio de ejemplos del SDK de AWS documentación. Consulte el ejemplo completo aquí.

```
/**
 * Stops the hosting an Amazon Lookout for Vision model. Returns when model has
 * stopped or if hosting fails.
 *
 * @param lfvClient   An Amazon Lookout for Vision client.
 * @param projectName The name of the project that contains the model that you
```

```
 *                    want to stop hosting.
 * @modelVersion The version of the model that you want to stop hosting.
 * @return ModelDescription The description of the model, which includes the
 *         model hosting status.
 */

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

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

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

        // Stop hosting the model.

        lfvClient.stopModel(stopModelRequest);

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

        ModelDescription modelDescription = null;

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

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

                switch (modelDescription.status()) {

                        case TRAINED:
                                logger.log(Level.INFO, "Model version {0} for
 project {1} has stopped.",
```

```
                                                    new Object[] { modelVersion,
 projectName });
                                finished = true;
                                break;

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

                                TimeUnit.SECONDS.sleep(60);

                                break;

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

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

                }

        } while (!finished);

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

        return modelDescription;

}
```

Detección de anomalías en una imagen

Para detectar anomalías en una imagen con un modelo entrenado de Amazon Lookout for Vision, se llama a la operación. DetectAnomalies El resultado de `DetectAnomalies` incluye una predicción booleana que clasifica la imagen como si contenga una o más anomalías y un valor de confianza para la predicción. Si el modelo es un modelo de segmentación de imágenes, el resultado también incluye una máscara de color que muestra las posiciones de los distintos tipos de anomalías.

Las imágenes que suministre a `DetectAnomalies` deben tener las mismas dimensiones de ancho y alto que las imágenes que utilizó para entrenar el modelo.

`DetectAnomalies`acepta imágenes en formato PNG o JPG. Recomendamos que las imágenes estén en el mismo formato de codificación y compresión que los utilizados para entrenar el modelo. Por ejemplo, si entrena el modelo con imágenes en formato PNG, llame `DetectAnomalies` con imágenes en formato PNG.

Antes de llamar a `DetectAnomalies`, primero debe iniciar su modelo con la operación `StartModel`. Para obtener más información, consulte Inicio del modelo Amazon Lookout for Vision. Se le cobrará por la cantidad de tiempo, en minutos, que funcione un modelo y por el número de unidades de detección de anomalías que utilice su modelo. Si ya no utiliza el modelo, utilice la operación `StopModel` para detenerlo. Para obtener más información, consulte Paro del modelo Amazon Lookout for Vision.

Temas

¿Llamando DetectAnomalies

Para llamar a `DetectAnomalies`, debe indicar lo siguiente:

- Project: el nombre del proyecto que incluye el modelo que quiere usar.
- ModelVersion— La versión del modelo que desea utilizar.

- ContentType— El tipo de imagen que quieres analizar. Los valores válidos son `image/png` (imágenes en formato PNG) e `image/jpeg` (imágenes en formato JPG).
- Body: los bytes binarios no codificados que representan la imagen.

  Las imágenes deben tener las mismas dimensiones que las imágenes que utilizó para entrenar el modelo.

En el siguiente ejemplo, se muestra cómo llamar a `DetectAnomalies`. Puede utilizar la respuesta de la función de los ejemplos de Python y Java para llamar a las funciones de Determinar si una imagen es anómala.

AWS CLI

Este AWS CLI comando muestra el resultado JSON de la operación `DetectAnomalies` CLI. Cambie los valores de los siguientes parámetros de entrada:

- `project name` con el nombre del proyecto que desea usar.
- `model version` con la versión del modelo que desea utilizar.
- `content type` con el tipo de imagen que desea utilizar. Los valores válidos son `image/png` (imágenes en formato PNG) e `image/jpeg` (imágenes en formato JPG).
- `file name` con la ruta de acceso y el nombre de archivo de la imagen que desea utilizar. Asegúrese de que el tipo de archivo coincida con el valor de `content-type`.

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

Python

Para ver el ejemplo de código completo, consulte GitHub.

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

```
        :param model_version: The version of the model that you want to use.
        :param photo: The photo that you want to analyze.
        :return: The DetectAnomalyResult object that contains the analysis results.
        """

        image_type = imghdr.what(photo)
        if image_type == "jpeg":
            content_type = "image/jpeg"
        elif image_type == "png":
            content_type = "image/png"
        else:
            logger.info("Invalid image type for %s", photo)
            raise ValueError(
                f"Invalid file format. Supply a jpeg or png format file: {photo}")


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

        return response['DetectAnomalyResult']
```

Java V2

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

```
         * @return DetectAnomalyResult The analysis result from DetectAnomalies.
         */

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

        // Get image bytes.

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

        // Get the image type. Can be image/jpeg or image/png.
        String contentType = getImageType(imageBytes);

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

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

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

        String prediction = "Prediction: Normal";

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

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

        // Log classification result.
        String photoPath = "File: " + photo;
```

```
        String[] imageLines = { photoPath, prediction, confidence };
        logger.log(Level.INFO, "Image: {0}\nAnomalous: {1}\nConfidence {2}",
imageLines);

        return result;

    }

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

        InputStream is = new BufferedInputStream(new ByteArrayInputStream(image));
        String mimeType = URLConnection.guessContentTypeFromStream(is);

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

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

Entendiendo la respuesta de DetectAnomalies

La respuesta `DetectAnomalies` varía según el tipo de modelo que se entrene (modelo de clasificación o modelo de segmentación). En ambos casos, la respuesta es un DetectAnomalyResultobjeto.

Modelo de clasificación

Si su modelo es un Ajuste de un modelo de clasificación de imágenes, la respuesta de `DetectAnomalies` contiene lo siguiente:

- IsAnomalous— Un indicador booleano de que la imagen contiene una o más anomalías.
- Confianza: la confianza que Amazon Lookout for Vision tiene en la precisión de la predicción `IsAnomalous` de anomalías (). `Confidence`es un valor de punto flotante entre 0 y 1. Un valor más alto indica una confianza más alta.

- Source: información sobre la imagen a la que se le ha pasado a `DetectAnomalies`.

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

Para determinar si una imagen es anómala, compruebe el `IsAnomalous` campo y confirme que el `Confidence` valor es lo suficientemente alto para sus necesidades.

Si observa que los valores de confianza devueltos por `DetectAnomalies` son demasiado bajos, considere la posibilidad de volver a entrenar el modelo. Para ver el código de ejemplo, consulte Clasificación.

Modelo de segmentación

Si su modelo es un Métricas del modelo de segmentación de imágenes, la respuesta incluye información de clasificación y segmentación, como una máscara de imagen y tipos de anomalías. La información de clasificación se calcula por separado de la información de segmentación y no se debe suponer que existe una relación entre ambas. Si no aparece información de segmentación en la respuesta, compruebe que tiene instalada la última versión del AWS SDK (si la estáAWS Command Line Interface utilizando). AWS CLI Para ver código de ejemplo, consulte Segmentación y Mostrar información de clasificación y segmentación.

- IsAnomalous(clasificación): indicador booleano que clasifica la imagen como normal o anómala.
- Confidence (clasificación): la confianza que Amazon Lookout for Vision tiene en la precisión de la predicción de anomalías (`IsAnomalous`). `Confidence` es un valor de punto flotante entre 0 y 1. Un valor más alto indica una confianza más alta.
- Source: información sobre la imagen a la que se le ha pasado a `DetectAnomalies`.
- AnomalyMask(segmentación): máscara de píxeles que cubre las anomalías encontradas en la imagen analizada. Puede haber varias anomalías en la imagen. El color de los mapas de máscaras indica el tipo de anomalía. Los colores de la máscara se corresponden con los colores

asignados a los tipos de anomalías en el conjunto de datos de entrenamiento. Para encontrar el tipo de anomalía a partir del color de una máscara, marca `Color` el `PixelAnomaly` campo de cada anomalía devuelta en la lista `Anomalies`. Para ver el código de ejemplo, consulte Mostrar información de clasificación y segmentación.

- Anomalies (segmentación): lista de las anomalías que se encuentran en la imagen. Cada anomalía incluye el tipo de anomalía () y la información de píxeles (`Name`). `PixelAnomaly` `TotalPercentageArea`es el porcentaje de área de la imagen que cubre la anomalía. `Color`es el color de la máscara de la anomalía.

  El primer elemento de la lista es siempre un tipo de anomalía que representa el fondo de la imagen (`BACKGROUND`) y no debe considerarse una anomalía. Amazon Lookout for Vision añade automáticamente el tipo de anomalía de fondo a la respuesta. No es necesario declarar un tipo de anomalía de fondo en su conjunto de datos.


```
{
    "DetectAnomalyResult": {
        "Source": {
            "Type": "direct"
        },
        "IsAnomalous": true,
        "Confidence": 0.9996814727783203,
        "Anomalies": [
            {
                "Name": "background",
                "PixelAnomaly": {
                    "TotalPercentageArea": 0.998999834060669,
                    "Color": "#FFFFFF"
                }
            },
            {
                "Name": "scratch",
                "PixelAnomaly": {
                    "TotalPercentageArea": 0.0004034999874420464,
                    "Color": "#7ED321"
                }
            },
            {
                "Name": "dent",
                "PixelAnomaly": {
                    "TotalPercentageArea": 0.0005966666503809392,
```

```
                    "Color": "#4DD8FF"
                }
            }
        ],
        "AnomalyMask": "iVBORw0....."
    }
}
```

Determinar si una imagen es anómala

Puede determinar si una imagen es anómala de diversas formas. El método elegido depende del caso de uso y el tipo de modelo. Las siguientes son posibles soluciones.

Temas

- Clasificación
- Segmentación

Clasificación

`IsAnomalous` clasifica una imagen como anómala; utilice el campo `Confidence` para decidir si la imagen es realmente anómala. Un valor más alto indica una confianza más alta. Por ejemplo, puede decidir que un producto es defectuoso solo si la confianza es superior al 80 %. Puede clasificar las imágenes analizadas mediante modelos de clasificación o mediante modelos de segmentación de imágenes.

Python

Para ver el ejemplo de código completo, consulte. GitHub

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

```
        reject = False

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

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

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

Java V2

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

        Boolean reject = false;

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

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

        if (Boolean.TRUE.equals(prediction.isAnomalous()) && prediction.confidence()
>= minConfidence) {
```

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

        return reject;

    }
```

Segmentación

Si su modelo es un modelo de segmentación de imágenes, puede utilizar la información de segmentación para determinar si una imagen contiene anomalías. También puede utilizar un modelo de segmentación de imágenes para clasificar las imágenes. Para ver un ejemplo de código que obtiene y muestra máscaras de imágenes, consulte Mostrar información de clasificación y segmentación

Área de anomalía

Utilice el porcentaje de cobertura (`TotalPercentageArea`) de una anomalía en la imagen. Por ejemplo, puede decidir que un producto es defectuoso si el área de la anomalía es superior al 1% de la imagen.

Python

Para ver el ejemplo de código completo, consulte GitHub.

```
    def reject_on_coverage(image, prediction, confidence_limit, anomaly_label,
coverage_limit):
        """
        Checks if the coverage area of an anomaly is greater than the coverage limit
and if
        the prediction confidence is greater than the confidence limit.
        :param image: The name of the image file that was analyzed.
        :param prediction: The DetectAnomalyResult object returned from
DetectAnomalies
        :param confidence_limit: The minimum acceptable confidence (float 0-1).
```

```
        :anomaly_label: The anomaly label for the type of anomaly that you want to
 check.
        :coverage_limit: The maximum acceptable percentage coverage of an anomaly
 (float 0-1).
        :return: True if the error condition indicates an anomaly, otherwise False.
        """

        reject = False

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

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

                    logger.info("%s", reject_info)

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

        return reject
```

Java V2

```
   public static Boolean rejectOnCoverage(String image, DetectAnomalyResult
 prediction, float minConfidence,
            String anomalyType, float maxCoverage) {
        /**
         * Rejects an image based on a maximum allowable coverage area for an
 anomaly
         * type.
```

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

       Boolean reject = false;

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

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

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

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

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

               }
           }
       }
```

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

        return reject;
    }
```

Número de tipos de anomalías

Utilice un recuento de los diferentes tipos de anomalías (Name) que se encuentran en la imagen. Por ejemplo, puede decidir que un producto es defectuoso si hay más de dos tipos de anomalías presentes.

Python

Para ver el ejemplo de código completo, consulte GitHub.

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

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

       reject = False

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

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

           if len (anomaly_types) > anomaly_types_limit:
```

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

                    logger.info("%s", reject_info)

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

Java V2

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

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

        Boolean reject = false;

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

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

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

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

        }

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

        return reject;
    }
```

Mostrar información de clasificación y segmentación

En este ejemplo se muestra la imagen analizada y se superponen los resultados del análisis. Si la respuesta incluye una máscara de anomalía, la máscara se muestra con los colores de los tipos de anomalías asociados.

Para mostrar la información de clasificación y segmentación de imágenes

1. Si aún no lo ha hecho, haga lo siguiente:

   a. Si aún no lo ha hecho, instale y configure el AWS CLI y el AWS SDKs. Para obtener más información, consulte Paso 4: Configura el AWS CLI y AWS SDKs.

   b. Entrenar el modelo.

   c. Inicie el modelo.

2. Asegúrese de que el usuario que llama a `DetectAnomalies` tenga acceso a la versión del modelo que desea utilizar. Para obtener más información, consulte Configuración de permisos de SDK.

3. Utilice el siguiente código.

Python

El siguiente código de ejemplo detecta anomalías en una imagen que usted proporciona. La línea de comandos de ejemplo acepta las siguientes opciones:

- `project`: el nombre del proyecto que desea usar.
- `version`: la versión del modelo dentro del proyecto que desea utilizar.
- `image`: la ruta y el archivo de un archivo de imagen local (formato JPEG o PNG).

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

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

from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)


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



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

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

        y_coordinate += text_height

        return y_coordinate

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

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

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

        draw = ImageDraw.Draw(image)

        y_coordinate = 0

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

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

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

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

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

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

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

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

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

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

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

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

           # Get images bytes for call to detect_anomalies.
           image_bytes = io.BytesIO()
           image.save(image_bytes, format=image.format)
           image_bytes = image_bytes.getvalue()

           # Analyze the image.
           response = lookoutvision_client.detect_anomalies(
               ProjectName=project_name,
               ContentType=image_type,
               Body=image_bytes,
               ModelVersion=model_version
           )
```

```
            # Overlay mask onto analyzed image.
            image_mask_bytes = response["DetectAnomalyResult"]["AnomalyMask"]
            image_mask = Image.open(io.BytesIO(image_mask_bytes))

            final_img = Image.blend(image, image_mask, 0.5) \
                if response["DetectAnomalyResult"]["IsAnomalous"] else image

            # Overlay analysis output on image.
            ShowAnomalies.draw_analysis_text(final_img, response)

            final_img.show()

        except ClientError as err:
            logger.info(format(err))
            raise


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

    parser.add_argument(
        "project", help="The project containing the model that you want to use."
    )
    parser.add_argument(
        "version", help="The version of the model that you want to use."
    )
    parser.add_argument(
        "image",
        help="The file that you want to analyze. "
        "Supply a local file path.",
    )


def main():
    """
    Entrypoint for anomaly detection example.
    """

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

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

        lookoutvision_client = session.client("lookoutvision")

        parser = argparse.ArgumentParser(usage=argparse.SUPPRESS)

        add_arguments(parser)

        args = parser.parse_args()

        # Analyze the image and show results.
        ShowAnomalies.show_anomaly_prediction(
            lookoutvision_client, args.project, args.version, args.image
        )

    except ClientError as err:
        print("A service error occured: " +
              format(err.response["Error"]["Message"]))
    except FileNotFoundError as err:
        print("The supplied file couldn't be found: " + err.filename)
    except ValueError as err:
        print("A value error occured. " + format(err))
    else:
        print("Successfully completed analysis.")


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

Java 2

El siguiente código de ejemplo detecta anomalías en una imagen que usted proporciona. La línea de comandos de ejemplo acepta las siguientes opciones:

- `project`: el nombre del proyecto que desea usar.
- `version`: la versión del modelo dentro del proyecto que desea utilizar.
- `image`: la ruta y el archivo de un archivo de imagen local (formato JPEG o PNG).

```
/*
```

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


package com.example.lookoutvision;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.services.lookoutvision.LookoutVisionClient;
import software.amazon.awssdk.services.lookoutvision.model.Anomaly;
import
 software.amazon.awssdk.services.lookoutvision.model.DetectAnomaliesRequest;
import
 software.amazon.awssdk.services.lookoutvision.model.DetectAnomaliesResponse;
import software.amazon.awssdk.services.lookoutvision.model.DetectAnomalyResult;
import
 software.amazon.awssdk.services.lookoutvision.model.LookoutVisionException;

import java.io.BufferedInputStream;
import java.io.ByteArrayInputStream;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.io.InputStream;
import java.net.URLConnection;

import java.text.NumberFormat;
import java.awt.*;
import java.awt.font.LineMetrics;
import java.awt.image.BufferedImage;
import javax.imageio.ImageIO;
import javax.swing.*;

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

// Finds anomalies on a supplied image.
public class ShowAnomalies extends JPanel {
/**
 * Finds and displays anomalies on a supplied image.
 */
```

```
   private static final long serialVersionUID = 1L;
   private transient BufferedImage image;
   private transient BufferedImage maskImage;
   private transient Dimension dimension;
   public static final Logger logger =
Logger.getLogger(ShowAnomalies.class.getName());

   // Constructor. Finds anomalies in a local image file.
   public ShowAnomalies(LookoutVisionClient lfvClient, String projectName,
String modelVersion,
           String photo) throws IOException, LookoutVisionException {

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

       maskImage = null;

       // Get image bytes and buffered image.
       InputStream sourceStream = new FileInputStream(new File(photo));
       SdkBytes imageSDKBytes = SdkBytes.fromInputStream(sourceStream);
       byte[] imageBytes = imageSDKBytes.asByteArray();
       ByteArrayInputStream inputStream = new
ByteArrayInputStream(imageSDKBytes.asByteArray());
       image = ImageIO.read(inputStream);

       // Get the image type. Can be image/jpeg or image/png.
       String contentType = getImageType(imageBytes);

       // Set the size of the window that shows the image.
       setWindowDimensions();

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

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

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

```
         */
        DetectAnomalyResult result = response.detectAnomalyResult();


        if (result.anomalyMask() != null){
            SdkBytes maskSDKBytes = result.anomalyMask();

            ByteArrayInputStream maskInputStream = new
ByteArrayInputStream(maskSDKBytes.asByteArray());
            maskImage = ImageIO.read(maskInputStream);
        }

        drawImageInfo(result);

    }

    // Sets window dimensions to 1/2 screen size, unless image is smaller.
    public void setWindowDimensions() {
        dimension = java.awt.Toolkit.getDefaultToolkit().getScreenSize();

        dimension.width = (int) dimension.getWidth() / 2;
        dimension.height = (int) dimension.getHeight() / 2;

        if (image.getWidth() < dimension.width || image.getHeight() <
dimension.height) {
            dimension.width = image.getWidth();
            dimension.height = image.getHeight();
        }
        setPreferredSize(dimension);

    }

    private int drawLine(Graphics2D g2d, String line, FontMetrics metrics, int
yPos, Color color) {
    /**
    * Draws a line of text at the spsecified y position and color.
    * confidence
    *
    * @param g2D The Graphics2D object for the image.
    * @param line The line of text to draw.
    * @param metrics The font information to use.
    * @param yPos The y position for the line of text.
    *
    * @return  The yPos for the next line of text.
```

```
    */

        int indent = 10;

        // Get text height, width, and descent.
        int textWidth = metrics.stringWidth(line);
        LineMetrics lm = metrics.getLineMetrics(line, g2d);
        int textHeight = (int) lm.getHeight();
        int descent = (int) lm.getDescent();

        int y2Pos = (yPos + textHeight) - descent;

        // Draw black rectangle.
        g2d.setColor(Color.BLACK);
        g2d.fillRect(indent, yPos, textWidth, textHeight);

        // Draw text.
        g2d.setColor(color);
        g2d.drawString(line, indent, y2Pos);

        yPos += textHeight;

        return yPos;

    }

    public void drawImageInfo(DetectAnomalyResult result) {
    /**
     * Draws the results from DetectAnomalies onto the output image.
     *
     * @param result The response from a call to
     *               DetectAnomalies.
     *
     */

        // Set up drawing.
        Graphics2D g2d = image.createGraphics();

        if (result.anomalyMask() != null){
            Composite composite = g2d.getComposite();
            g2d.setComposite(AlphaComposite.SrcOver.derive(0.5f));
            int x = (image.getWidth() - maskImage.getWidth()) / 2;
            int y = (image.getHeight() - maskImage.getHeight()) / 2;
            g2d.drawImage(maskImage, x, y, null);
```

```
            // Set composite for overlaying text.
            g2d.setComposite(composite);
        }

        //Calculate font size based on arbitary 32 pixel image width.
        int fontSize = (image.getWidth() / 32);


        g2d.setFont(new Font("Tahoma", Font.PLAIN, fontSize));
        Font font = g2d.getFont();
        FontMetrics metrics = g2d.getFontMetrics(font);

        // Get classification information.

        String prediction = "Prediction: Normal";

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

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

        // Draw classification information.
        int yPos = 0;

        yPos = drawLine(g2d, "Classification:", metrics, yPos, Color.WHITE);
        yPos = drawLine(g2d, prediction, metrics, yPos, Color.WHITE);
        yPos = drawLine(g2d, confidence, metrics, yPos, Color.WHITE);

        // Draw segmentation info.
        yPos = drawLine(g2d, "Segmentation:", metrics, yPos, Color.WHITE);

        // Ignore background label, so size must be > 1
        if (result.anomalies().size() > 1) {
            for (Anomaly anomaly : result.anomalies()) {
                if (anomaly.name().equals("background"))
                    continue;
                String label = String.format("Anomaly: %s. Area: %s",
anomaly.name(),
```

```
defaultFormat.format(anomaly.pixelAnomaly().totalPercentageArea()));
                Color anomalyColor =
Color.decode((anomaly.pixelAnomaly().color()));
                yPos = drawLine(g2d, label, metrics, yPos, anomalyColor);

            }

        } else {
            drawLine(g2d, "None found.", metrics, yPos, Color.WHITE);

        }

        g2d.dispose();

    }

    @Override
    public void paintComponent(Graphics g)
    /**
     * Draws the image and analysis results.
     *
     * @param g The Graphics context object for drawing.
     *          DetectAnomalies.
     *
     */
    {

        Graphics2D g2d = (Graphics2D) g; // Create a Java2D version of g.

        // Draw the image.
        g2d.drawImage(image, 0, 0, dimension.width, dimension.height, this);

    }

    // Gets the image mime type. Supported formats are image/jpeg and image/png.

    private String getImageType(byte[] image) throws IOException
    /**
     * Gets the file type of a supplied image. Raises an exception if the image
     * isn't compatible with with Amazon Lookout for Vision.
     *
     * @param image The image that you want to check.
     *
```

```
     * @return String The type of the image.
     */

   {
       InputStream is = new BufferedInputStream(new
ByteArrayInputStream(image));
       String mimeType = URLConnection.guessContentTypeFromStream(is);

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

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

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

       String photo = null;
       String projectName = null;
       String modelVersion = null;

       final String USAGE = "\n" +
               "Usage:\n" +
               "    DetectAnomalies <project> <version> <image> \n\n" +
               "Where:\n" +
               "    project - The Lookout for Vision project.\n\n" +
               "    version - The version of the model within the project.\n\n"
+
               "    image - The path and filename of a local image. \n\n";

       try {

           if (args.length != 3) {
               System.out.println(USAGE);
               System.exit(1);
           }

           projectName = args[0];
           modelVersion = args[1];
           photo = args[2];
```

```
            ShowAnomalies panel = null;

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

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

            // Create frame and panel.
            JFrame frame = new JFrame(photo);
            frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

            panel = new ShowAnomalies(lfvClient, projectName, modelVersion,
photo);

            frame.setContentPane(panel);
            frame.pack();
            frame.setVisible(true);


        } catch (LookoutVisionException lfvError) {
            logger.log(Level.SEVERE, "Lookout for Vision client error: {0}:
{1}",
                    new Object[] { lfvError.awsErrorDetails().errorCode(),
                            lfvError.awsErrorDetails().errorMessage() });
            System.out.println(String.format("lookout for vision client error:
%s", lfvError.getMessage()));
            System.exit(1);

        } catch (FileNotFoundException fileError) {
            logger.log(Level.SEVERE, "Could not find file: {0}",
fileError.getMessage());
            System.out.println(String.format("Could not find file: %s",
fileError.getMessage()));
            System.exit(1);

        } catch (IOException ioError) {
            logger.log(Level.SEVERE, "IO error {0}", ioError.getMessage());
            System.out.println(String.format("IO error: %s",
ioError.getMessage()));
            System.exit(1);
        }

    }
```

```
}
```

4. Si no planea seguir usando su modelo, deténgalo.

Búsqueda de anomalías con una función AWS Lambda

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 DetectAnomalies. La función analiza una imagen proporcionada y devuelve una clasificación para detectar la presencia de anomalías en esa 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 equipo local.

Temas

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 `DetectAnomalies` 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 proyecto y la versión del modelo de Amazon Lookout for Vision que desea que utilice su 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
PROJECT_NAME	El proyecto Lookout for Vision que contiene el modelo que desea utilizar.
MODEL_VERSION	La versión del modelo que desea utilizar.
CONFIDENCE	El valor mínimo (0-100) de confianza del modelo en la predicción de una anomalía. Si la confianza es inferior, la clasificación se considera normal.

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": "lookoutvision:DetectAnomalies",
            "Resource": "*",
```

```
            "Effect": "Allow",
            "Sid": "DetectAnomaliesAccess"
        }
    ]
}
```

12. Elija Next (Siguiente).
13. En Detalles de la política, introduce un nombre para la política, como DetectAnomalies-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 `DetectAnomalies` 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, añade código Python a la función de Lambda mediante el editor de código de la consola de Lambda. El código analiza una imagen proporcionada `DetectAnomalies` y devuelve

una clasificación (verdadera si la imagen es anómala, falsa si la imagen es normal). 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 Amazon Lookout for Vision
 model.
"""
import base64
import imghdr
from os import environ
from io import BytesIO
import logging
import boto3

from botocore.exceptions import ClientError


logger = logging.getLogger(__name__)


# Get the model and confidence.
project_name = environ['PROJECT_NAME']
model_version = environ['MODEL_VERSION']
min_confidence = int(environ.get('CONFIDENCE', 50))

lookoutvision_client = boto3.client('lookoutvision')
```

```
def lambda_handler(event, context):
    """
    Lambda handler function
    param: event: The event object for the Lambda function.
    param: context: The context object for the lambda function.
    return: The labels found in the image passed in the event
    object.
    """

    try:

        file_name = ""

        # Determine image source.
        if 'image' in event:
            # Decode the encoded image
            image_bytes = event['image'].encode('utf-8')
            img_b64decoded = base64.b64decode(image_bytes)
            image_type = get_image_type(img_b64decoded)
            image = BytesIO(img_b64decoded)
            file_name = event['filename']

        elif 'S3Object' in event:
            bucket = boto3.resource('s3').Bucket(event['S3Object']['Bucket'])
            image_object = bucket.Object(event['S3Object']['Name'])
            image = image_object.get().get('Body').read()
            image_type = get_image_type(image)
            file_name = f"s3://{event['S3Object']['Bucket']}/{event['S3Object']
['Name']}"

        else:
            raise ValueError(
                'Invalid image source. Only base 64 encoded image bytes or images
 in S3 buckets are supported.')

        # Analyze the image.
        response = lookoutvision_client.detect_anomalies(
            ProjectName=project_name,
            ContentType=image_type,
            Body=image,
            ModelVersion=model_version)

        reject = reject_on_classification(
```

```
            response['DetectAnomalyResult'],
 confidence_limit=float(environ['CONFIDENCE'])/100)

        status = "anomalous" if reject else "normal"

        lambda_response = {
            "statusCode": 200,
            "body": {
                "Reject": reject,
                "RejectMessage": f"Image {file_name} is {status}."
            }
        }

    except ClientError as err:
        error_message = f"Couldn't analyze {file_name}. " + \
            err.response['Error']['Message']

        lambda_response = {
            'statusCode': 400,
            'body': {
                "Error": err.response['Error']['Code'],
                "ErrorMessage": error_message,
                "Image": file_name
            }
        }
        logger.error("Error function %s: %s",
                     context.invoked_function_arn, error_message)

    except ValueError as val_error:

        lambda_response = {
            'statusCode': 400,
            'body': {
                "Error": "ValueError",
                "ErrorMessage": format(val_error),
                "Image": event['filename']
            }
        }
        logger.error("Error function %s: %s",
                     context.invoked_function_arn, format(val_error))

    return lambda_response

def get_image_type(image):
```

```
    """
    Gets the format of the image. Raises an error
    if the type is not PNG or JPEG.
    :param image: The image that you want to check.
    :return The type of the image.

    """
    image_type = imghdr.what(None, image)

    if image_type == "jpeg":
        content_type = "image/jpeg"
    elif image_type == "png":
        content_type = "image/png"
    else:
        logger.info("Invalid image type")
        raise ValueError(
            "Invalid file format. Supply a jpeg or png format file.")
    return content_type


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

    reject = False

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

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

4. 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. Si aún no lo ha hecho, haga lo siguiente:

   a. Asegúrese de que el usuario que usa el código de cliente tiene permiso `lambda:InvokeFunction`. Puede usar los siguientes permisos.

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

b. Instale y configure el AWS SDK para Python. Para obtener más información, consulte Paso 4: Configura el AWS CLI y AWS SDKs.

c. Inicie el modelo que eligió en el paso 7 de Paso 1: Crear una AWS Lambda función (consola).

2. 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: Shows how to call the anomaly detection
AWS Lambda function.
"""
from botocore.exceptions import ClientError

import argparse
import logging
import base64
```

```
import json
import boto3
from os import environ

logger = logging.getLogger(__name__)

def analyze_image(function_name, image):
    """
    Analyzes an image with an AWS Lambda function.
    :param image: The image that you want to analyze.
    :return The status and classification result for
    the image analysis.
    """

    lambda_client = boto3.client('lambda')

    lambda_payload = {}

    if image.startswith('s3://'):
        logger.info("Analyzing image from S3 bucket: %s", image)
        bucket, key = image.replace("s3://", "").split("/", 1)
        s3_object = {
            'Bucket': bucket,
            'Name': key
        }
        lambda_payload = {"S3Object": s3_object}

    # Call the lambda function with the image.
    else:
        with open(image, 'rb') as image_file:
            logger.info("Analyzing local image image: %s ", image)
            image_bytes = image_file.read()
            data = base64.b64encode(image_bytes).decode("utf8")
            lambda_payload = {"image": data, "filename": image}

    response = lambda_client.invoke(FunctionName=function_name,
                                    Payload=json.dumps(lambda_payload))
    return json.loads(response['Payload'].read().decode())


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

```
    """

    parser.add_argument(
        "function", help="The name of the AWS Lambda function "
        "that you want to use to analyze the image.")
    parser.add_argument(
        "image", help="The local image that you want to analyze.")


def main():
    """
    Entrypoint for script.
    """
    try:
        logging.basicConfig(level=logging.INFO,
                            format="%(levelname)s: %(message)s")

        # Get command line arguments.
        parser = argparse.ArgumentParser(usage=argparse.SUPPRESS)
        add_arguments(parser)
        args = parser.parse_args()

        # Analyze image and display results.

        result = analyze_image(args.function, args.image)

        status = result['statusCode']

        if status == 200:
            classification = result['body']
            print(f"classification: {classification['Reject']}")
            print(f"Message: {classification['RejectMessage']}")
        else:
            print(f"Error: {result['statusCode']}")
            print(f"Message: {result['body']}")

    except ClientError as error:
        logging.error(error)
        print(error)


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

3. Ejecute el código. En el argumento de la línea de comandos, indique el nombre de la función de Lambda y la ruta a la imagen local que desee analizar. Por ejemplo:

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

   Si se ejecuta correctamente, el resultado es una clasificación de las anomalías encontradas en la imagen. Si no devuelve ninguna clasificación, considere la posibilidad de reducir el valor de confianza que eligió en el paso 7 de Paso 1: Crear una AWS Lambda función (consola).
4. 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.

Uso del modelo Amazon Lookout for Vision en un dispositivo de borde

Puede utilizar su modelo Amazon Lookout for Vision en dispositivos periféricos gestionados AWS IoT Greengrass Version 2 por. AWS IoT Greengrass es un servicio de nube y tiempo de ejecución periférica del Internet de las Cosas (IoT) de código abierto. Puede usarlo para crear, implementar y administrar aplicaciones de IoT en sus dispositivos. Para obtener más información, consulte AWS IoT Greengrass.

Implementa los mismos modelos de Amazon Lookout for Vision que ha entrenado en la nube en dispositivos de borde compatibles con AWS IoT Greengrass V2 . A continuación, puede usar el modelo implementado para realizar la detección de anomalías in situ, como en una fábrica, sin transmitir datos continuamente a la nube. De esta forma, puede minimizar los costes de ancho de banda y detectar anomalías a nivel local mediante el análisis de imágenes en tiempo real.

> **Tip**
>
> Antes de implementar un modelo de Lookout for Vision AWS IoT Greengrass con, le recomendamos que lea AWS IoT Greengrass Version 2 la guía para desarrolladores. Para obtener más información, consulte ¿Qué es AWS IoT Greengrass?.

Para utilizar un modelo Lookout for Vision en AWS IoT Greengrass V2 un dispositivo principal, debe implementar el modelo y el software de soporte como componentes del dispositivo principal. Un componente es un módulo de software, como un modelo Lookout for Vision, que se ejecuta en un dispositivo principal de Greengrass. Hay dos tipos de componentes. Un componente personalizado es un componente que usted crea y al que solo puede acceder usted. También se denomina componente privado. Un componente AWS suministrado es un componente prediseñado que AWS proporciona. También se denomina componente público. Para obtener más información, consulte https://docs.aws.amazon.com/greengrass/v2/developerguide/public-components.html.

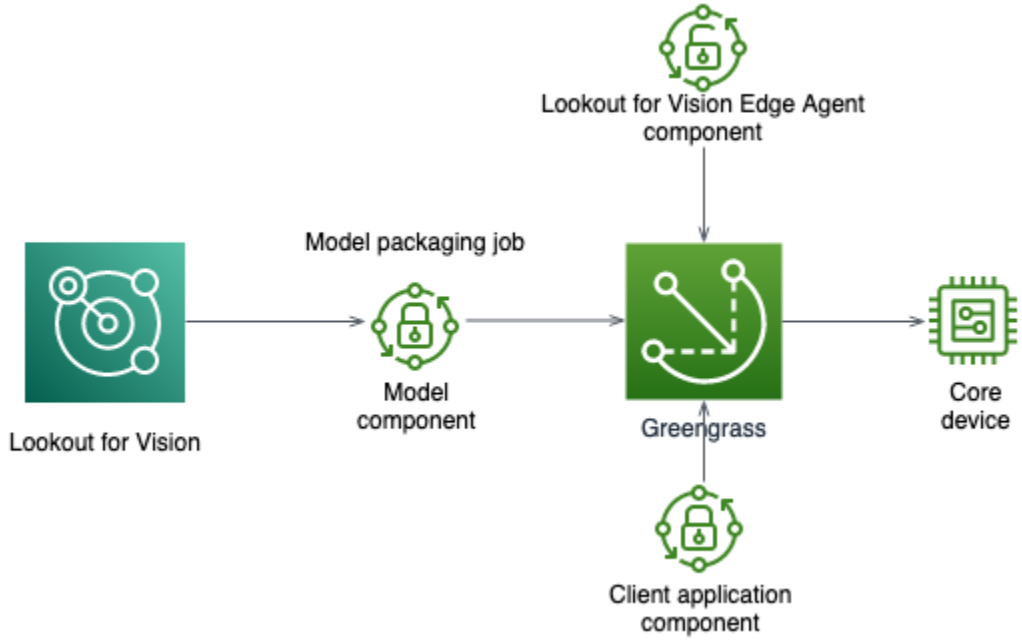
Los componentes que se implementan en un dispositivo principal para un modelo de Lookout for Vision y el software de soporte son:

- Componente del modelo. Un componente personalizado que contiene su modelo Lookout for Vision. Para crear el componente del modelo, utilice Lookout for Vision para crear un trabajo

de empaquetado de modelos. Un trabajo de empaquetado de modelos crea un componente para el modelo y lo hace disponible como un componente personalizado en su interior AWS IoT Greengrass V2. Para obtener más información, consulte Empaquetado del modelo Amazon Lookout for Vision.

- Componente de aplicación cliente. Un componente personalizado que se crea y que implementa el código según las necesidades de su empresa. Por ejemplo, encontrar placas de circuito anómalas a partir de imágenes tomadas después del ensamblaje. Para obtener más información, consulte Escribiendo el componente de la aplicación cliente.
- Componente Amazon Lookout for Vision Edge Agent. Un componente AWS suministrado que proporciona una API para usar y administrar el modelo. Por ejemplo, el código del componente de la aplicación cliente puede usar la `DetectAnomalies` API para detectar anomalías en las imágenes. El componente Lookout for Vision Edge Agent es una dependencia del componente del modelo. Se instala automáticamente en el dispositivo principal al implementar el componente del modelo. Para obtener más información, consulte Amazon Lookout for Vision Edge Agent API.

Tras crear el componente del modelo y el componente de la aplicación del cliente, puede AWS IoT Greengrass V2 utilizarlos para implementar los componentes y las dependencias en el dispositivo principal. Para obtener más información, consulte Implementación de sus componentes en un dispositivo.



> **Important**
>
> Las predicciones que realiza su modelo con `DetectAnomalies` en un dispositivo principal pueden diferir de las predicciones realizadas con el mismo modelo alojado en la nube. Le recomendamos que pruebe el modelo en un dispositivo principal antes de usarlo en un entorno de producción.
>
> Para reducir los desajustes de predicción entre los modelos alojados en dispositivos y los modelos alojados en la nube, le recomendamos aumentar el número de imágenes normales y anómalas en su conjunto de datos de entrenamiento. No recomendamos reutilizar las imágenes existentes para aumentar el tamaño del conjunto de datos de entrenamiento.

Implementación de un modelo y un componente de aplicación cliente en un dispositivo AWS IoT Greengrass Version 2 principal

El procedimiento para implementar un modelo y un componente de aplicación cliente de Amazon Lookout for Vision en AWS IoT Greengrass Version 2 un dispositivo principal es el siguiente:

1. Configure sus dispositivos principales con AWS IoT Greengrass Version 2.
2. Cree un trabajo de empaquetado de modelos con Lookout for Vision. El trabajo crea el componente del modelo.
3. Escriba un componente de aplicación cliente. El componente implementa su lógica empresarial.
4. Implemente el componente del modelo y el componente de la aplicación del cliente en el dispositivo principal mediante AWS IoT Greengrass V2.

Una vez implementados los componentes y las dependencias en el dispositivo principal, puede usar el modelo en el dispositivo principal.

> **Note**
>
> Debes usar la misma AWS región y AWS cuenta para crear e implementar tu modelo de Lookout for Vision y el componente de aplicación cliente.

AWS IoT Greengrass Version 2 requisitos principales del dispositivo

Para utilizar un modelo de Amazon Lookout for Vision en AWS IoT Greengrass Version 2 un dispositivo principal, el modelo tiene varios requisitos del dispositivo principal.

Temas

Dispositivos, arquitecturas de chips y sistemas operativos probados

Esperamos que Amazon Lookout for Vision funcione en el siguiente hardware:

- Arquitecturas de la CPU
  - X86_64 (versión de 64 bits del conjunto de instrucciones x86)
  - Aarch64 (CPU de 64 bits) ARMv8
- (Solo inferencia acelerada por GPU) Acelerador de GPU NVIDIA con suficiente capacidad de memoria (al menos 6 GB para un modelo en ejecución).

El equipo de Amazon Lookout for Vision ha probado los modelos de Lookout for Vision en los siguientes dispositivos, arquitecturas de chips y sistemas operativos.

Dispositivos

Dispositivo	Sistema operativo	Arquitectura	Acelerador	Opciones del compilador
jetson_xavier (NVIDIA® Jetson AGX Xavier)	Linux	Aarch64	NVIDIA	`{"gpu-code": "sm_72", "trt-ver": "7.1.3",`

Dispositivo	Sistema operativo	Arquitectura	Acelerador	Opciones del compilador

				`"cuda-ver": "10.2"}`
				
				`{"gpu-code": "sm_72", "trt-ver" : "8.2.1", "cuda-ver": "10.2"}`

g4dn.xlarge (EC2 instancias (G4) con NVIDIA T4 Tensor Core) GPUs	Linux	X86_64/X86-64	NVIDIA	`{"gpu-code": "sm_75", "trt-ver" : "7.1.3", "cuda-ver": "10.2"}`
g5.xlarge (instancias (G5) con NVIDIA A10G Tensor Core) EC2 GPUs	Linux	X86_64/X86-64	NVIDIA	`{"gpu-code": "sm_80", "trt-ver" : "8.2.0", "cuda-ver": "11.2"}`
c5.2xlarge (instancias Amazon EC2 C5)	Linux	X86_64/X86-64	CPU	`{"mcpu": "core-avx2"}`

Memoria y almacenamiento del dispositivo principal

Para ejecutar un único modelo y el agente Amazon Lookout for Vision Edge, el dispositivo principal tiene los siguientes requisitos de memoria y almacenamiento. Es posible que necesite más memoria y almacenamiento para el componente de la aplicación cliente.

- Almacenamiento: 1,5 GB como mínimo.
- Memoria: al menos 6 GB para un modelo en ejecución.

Software necesario

Un dispositivo principal requiere el siguiente software.

Dispositivos Jetson

Si su dispositivo principal es un dispositivo Jetson, necesitará instalar el siguiente software en el dispositivo principal.

Software	Versiones compatibles
El SDK de Jetpack	De 4.4 a 4.6.1
Python y entorno virtual Python para Lookout for Vision Edge Agent versión 1.x	3.8 o 3.9

Hardware de X86;

Si su dispositivo principal utiliza hardware x86, necesitará tener instalado el siguiente software en el dispositivo principal.

Inferencia con CPU

Software	Versiones compatibles
Python y entorno virtual Python para Lookout for Vision Edge Agent versión 1.x	3.8 o 3.9

Inferencias aceleradas por GPU

Las versiones del software varían según la microarquitectura de la GPU NVIDIA que utilices.

GPU NVIDIA con microarquitectura anterior a Ampere (la capacidad de cómputo es inferior a 8.0)

Software necesario para una GPU NVIDIA con una microarquitectura anterior a Ampere (capacidad de procesamiento inferior a 8.0). El `gpu-code` debe ser inferior a `sm_80`.

Software	Versiones compatibles
nvidia-cuda	10.2
NVIDIA TensorRT	Al menos 7.1.3 y menos de 8.0.0
Python y entorno virtual Python para Lookout for Vision Edge Agent versión 1.x	3.8 o 3.9

GPU NVIDIA con microarquitectura Ampere (capacidad de cómputo 8.0)

Software necesario para una GPU NVIDIA con la microarquitectura Ampere (la capacidad de cálculo es de 8.0). El `gpu-code` debe ser `sm_80`.

Software	Versiones compatibles
nvidia-cuda	11.2
NVIDIA TensorRT	8.2.0
Python y entorno virtual Python para Lookout for Vision Edge Agent versión 1.x	3.8 o 3.9

Configuración del dispositivo AWS IoT Greengrass Version 2 principal

Amazon Lookout for Vision se AWS IoT Greengrass Version 2 utiliza para simplificar la implementación del componente modelo, el componente de agente de Amazon Lookout for Vision

Edge y el componente de aplicación de cliente en su dispositivo principal. AWS IoT Greengrass V2 Para obtener información sobre los dispositivos y el hardware que puede utilizar, consulte AWS IoT Greengrass Version 2 requisitos principales del dispositivo.

Configuración del dispositivo principal

Use la siguiente información para configurar su dispositivo principal.

Configurar el dispositivo principal

1. Configurar las bibliotecas de GPU. No realice este paso si no utiliza la inferencia acelerada por GPU.

   a. Compruebe que tiene una GPU compatible con CUDA. Para obtener más información, consulte Verificar que tiene una GPU compatible con CUDA.

   b. Configure CUDA, cuDNN y TensorRT en el dispositivo de la siguiente manera:

      - Si utiliza un dispositivo Jetson, instale las JetPack versiones 4.4 a 4.6.1. Para obtener más información, consulte JetPack Archivo.
      - Si utiliza un hardware basado en x86 y la microarquitectura de su GPU NVIDIA es anterior a Ampere (la capacidad de procesamiento es inferior a 8.0), haga lo siguiente:
        1. Configure la versión 10.2 de CUDA siguiendo las instrucciones de la Guía de instalación de NVIDIA CUDA para Linux.
        2. Instale cuDNN siguiendo las instrucciones de la documentación de NVIDIA cuDNN.
        3. Configure TensorRT (versión 7.1.3 o posterior, pero anterior a la 8.0.0) siguiendo las instrucciones de la DOCUMENTACIÓN DE NVIDIA TENSORRT.
      - Si utiliza un hardware basado en x86 y la microarquitectura de su GPU NVIDIA es Ampere (la capacidad de procesamiento es 8.0), haga lo siguiente:
        1. Configure CUDA (versión 11.2) siguiendo las instrucciones de la Guía de instalación de NVIDIA CUDA para Linux.
        2. Instale cuDNN siguiendo las instrucciones de la documentación de NVIDIA cuDNN.
        3. Configure TensorRT (versión 8.2.0) siguiendo las instrucciones de la DOCUMENTACIÓN DE TENSORRT de NVIDIA.

2. Instale el software AWS IoT Greengrass Version 2 principal en su dispositivo principal. Para obtener más información, consulte Instalación de Software básico de AWS IoT Greengrass en la AWS IoT Greengrass Version 2 Guía para desarrolladores.

3. Para leer desde el bucket de Amazon S3 que almacena el modelo, asocie el permiso al rol de IAM (rol de intercambio de token) que creó durante la AWS IoT Greengrass Version 2 configuración. Para obtener más información, consulte Permitir el acceso a los buckets de S3 para los artefactos de los componentes.
4. En la línea de comandos, introduzca el siguiente comando para instalar Python y un entorno virtual de Python en el dispositivo principal.

```
sudo apt install python3.8 python3-venv python3.8-venv
```

5. Utilice el siguiente comando para añadir el usuario de Greengrass al grupo de vídeo. Esto permite que los componentes desplegados por Greengrass accedan a la GPU:

```
sudo usermod -a -G video ggc_user
```

6. (Opcional) Si quiere llamar a la API de agente de Lookout for Vision Edge desde otro usuario, añada el usuario necesario al `ggc_group`. Esto permite al usuario comunicarse con el agente Lookout for Vision Edge a través del socket de dominio Unix:

```
sudo usermod -a -G ggc_group $(whoami)
```

Empaquetado del modelo Amazon Lookout for Vision

Un trabajo de empaquetado de modelos empaqueta un modelo de Amazon Lookout for Vision como componente del modelo.

Para crear un trabajo de empaquetado de modelos, usted elige el modelo que desea empaquetar y proporciona la configuración para el componente del modelo que crea el trabajo. Solo puede empaquetar un modelo que se haya entrenado correctamente.

Puedes usar la consola AWS o el SDK de Lookout for Vision para crear el trabajo de empaquetado del modelo. También puede obtener información sobre los trabajos de empaquetado de modelos que cree. Para obtener más información, consulte Obtención de información sobre trabajos de empaquetado de modelos. Puede usar la AWS IoT Greengrass V2 consola o el AWS SDK para implementar los componentes en el dispositivo AWS IoT Greengrass Version 2 principal.

Temas

- Configuración del paquete

- Empaquetado del modelo (consola)
- Empaquetado del modelo (SDK)
- Obtención de información sobre trabajos de empaquetado de modelos

Configuración del paquete

Utilice la siguiente información para decidir la configuración del paquete para su trabajo de empaquetado de modelos.

Para crear un trabajo de empaquetado de modelos, consulte Empaquetado del modelo (consola) o Empaquetado del modelo (SDK).

Temas

- Hardware de destino
- Configuración de los componentes

Hardware de destino

Puede elegir un dispositivo de destino o una plataforma de destino para su modelo, pero no ambos. Para obtener más información, consulte Dispositivos, arquitecturas de chips y sistemas operativos probados.

Dispositivo de destino

El dispositivo de destino del modelo, como NVIDIA® Jetson AGX Xavier. No tiene que especificar opciones del compilador.

Plataforma de destino

Amazon Lookout for Vision admite las siguientes configuraciones de plataforma:

- Arquitecturas X86_64 (versión de 64 bits del conjunto de instrucciones x86) y Aarch64 (CPU de 64 bits). ARMv8
- Sistema operativo Linux.
- Inferencia mediante aceleradores NVIDIA o de CPU.

Debe especificar las opciones de compilador correctas para su plataforma de destino.

Opciones del compilador

Las opciones del compilador le permiten especificar la plataforma de destino para su dispositivo principal AWS IoT Greengrass Version 2 . Actualmente puede especificar las opciones de compilación siguientes.

Acelerador NVIDIA

- `gpu-code`: especifica el código de GPU del dispositivo principal que ejecuta el componente del modelo.
- `trt-ver`: especifica la versión de TensorRT en formato x.y.z.
- `cuda-ver`: especifica la versión CUDA en formato x.y.

Acelerador de CPU

- (Opcional) `mcpu`: especifica el conjunto de instrucciones. Por ejemplo, `core-avx2`. Si no proporciona un valor, Lookout for Vision usará el valor `core-avx2`.

Las opciones se especifican en formato JSON. Por ejemplo:

```
{"gpu-code": "sm_75", "trt-ver": "7.1.3", "cuda-ver": "10.2"}
```

Para obtener más ejemplos, consulte Dispositivos, arquitecturas de chips y sistemas operativos probados.

Configuración de los componentes

El trabajo de empaquetado de modelos crea un componente del modelo que contiene el modelo. El trabajo crea artefactos que se AWS IoT Greengrass V2 utilizan para implementar el componente del modelo en el dispositivo principal.

No puede crear un componente de modelo con el mismo nombre y la misma versión de componente que un componente existente.

Nombre del componente

Nombre del componente del modelo que Lookout for Vision crea durante el empaquetado del modelo. El nombre del componente que especifique se muestra en la AWS IoT Greengrass V2

consola. El nombre del componente se utiliza en la receta que se crea para el componente de la aplicación cliente. Para obtener más información, consulte Creación del componente de la aplicación cliente.

Descripción del componente

(Opcional) Una descripción para el modelo.

Versión del componente

Un número de versión del componente del modelo. Puede aceptar el número de versión predeterminado o elegir el suyo propio. El número de versión debe seguir el sistema semántico de numeración de versiones: major.minor.patch. Por ejemplo, la versión 1.0.0 representa la primera versión principal de un componente. Para obtener más información, consulte Semantic Versioning 2.0.0. Si no proporciona un valor, Lookout for Vision usará el número de versión de su modelo para generar una versión para usted.

Ubicación del componente

La ubicación de Amazon S3 donde desea que el trabajo de empaquetado de modelos guarde los artefactos de los componentes del modelo. El bucket de Amazon S3 debe estar en la misma región de AWS y en la misma cuenta de AWS en la que utiliza AWS IoT Greengrass Version 2. Si desea crear un bucket de Amazon S3, consulte Creación de un bucket.

Etiquetas

Puede identificar, organizar, buscar y filtrar sus componentes mediante etiquetas. Cada etiqueta es una marca que consta de una clave y un valor definidos por el usuario. Las etiquetas se adjuntan al componente del modelo cuando el trabajo de empaquetado de modelos crea el componente del modelo en Greengrass. Un componente es un recurso de AWS IoT Greengrass V2. Las etiquetas no están adjuntas a ninguno de sus recursos de Lookout for Vision, como sus modelos. Para obtener más información, consulte Etiquetado de recursos de AWS.

Empaquetado del modelo (consola)

Puede crear un trabajo de empaquetado de modelos con la consola de Amazon Lookout for Vision.

Para obtener más información acerca de una configuración de empaquetado, consulte Configuración del paquete.

Para empaquetar un modelo (consola)

1. Cree un bucket de Amazon S3, o reutilice uno existente, que Lookout for Vision utiliza para almacenar los artefactos del trabajo de empaquetado (componente del modelo).
2. Abre la consola https://console.aws.amazon.com/lookoutvision/Amazon Lookout for Vision en.
3. Elija Comenzar.
4. En el panel de navegación izquierdo, elija Proyectos.
5. En la sección Proyectos, elija el proyecto que contiene el modelo que desea empaquetar.
6. En el panel de navegación izquierdo, en el nombre del proyecto, elija Paquetes de modelos de borde.
7. En la sección Trabajos de empaquetado de modelos, elija Crear un trabajo de empaquetado de modelos.
8. Introduzca la configuración del paquete. Para obtener más información, consulte Configuración del paquete.
9. Seleccione Crear trabajo de empaquetado de modelos.
10. Espere hasta que finalice el trabajo de empaquetado. El trabajo finaliza cuando el estado del trabajo es Éxito.
11. Elija el trabajo de empaquetado en la sección Trabajos de empaquetado de modelos.
12. Elija Continuar con la implementación en Greengrass para continuar con la implementación del componente de su modelo en AWS IoT Greengrass Version 2. Para obtener más información, consulte Implementación de sus componentes en un dispositivo.

Empaquetado del modelo (SDK)

Para empaquetar un modelo como un componente del modelo, debe crear un trabajo de empaquetado de modelos. Para crear un trabajo de empaquetado modelo, se llama StartModelPackagingJobAPI. El trabajo por lotes puede tardar un tiempo en completarse. Para averiguar el estado actual, llama DescribeModelPackagingJoby comprueba el `Status` campo de la respuesta.

Para obtener más información acerca de una configuración de empaquetado, consulte Configuración del paquete.

El siguiente procedimiento muestra cómo iniciar un trabajo de empaquetado mediante la AWS CLI. Puede empaquetar el modelo para una plataforma de destino o un dispositivo de destino. Para ver un ejemplo de código Java, consulte StartModelPackagingJob.

Para empaquetar el 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: Configura el AWS CLI y AWS SDKs.
2. Asegúrese de contar con los permisos correctos para iniciar un trabajo de empaquetado de modelos. Para obtener más información, consulte StartModelPackagingJob.
3. Utilice los siguientes comandos CLI para empaquetar el modelo para un dispositivo de destino o una plataforma de destino.

   Target platform

   El siguiente comando CLI muestra cómo empaquetar un modelo para una plataforma de destino con un acelerador NVIDIA.

   Cambie los siguientes valores:

   - `project_name` al nombre del proyecto que contiene el modelo que desea empaquetar.
   - `model_version` a la versión del modelo que desea empaquetar.
   - (Opcional) `description` a una descripción de su trabajo de empaquetado de modelos.
   - `architecture`a la arquitectura (`ARM64`o`X86_64`) del dispositivo AWS IoT Greengrass Version 2 principal en el que se ejecuta el componente del modelo.
   - `gpu_code` al código de la GPU del dispositivo principal en el que se ejecuta el componente del modelo.
   - `trt_ver` a la versión de TensorRT que tiene instalada en su dispositivo principal.
   - `cuda_ver` a la versión de CUDA que tiene instalada en su dispositivo principal.
   - `component_name`al nombre del componente del modelo en el que desee crear AWS IoT Greengrass V2.
   - (Opcional) `component_version` a una versión del componente del modelo que crea el trabajo de empaquetado. Utilice el formato `major.minor.patch`. Por ejemplo, la 1.0.0 representa la primera versión principal de un componente.
   - `bucket` al bucket de Amazon S3, donde el trabajo de empaquetado almacena los artefactos de los componentes del modelo.

- `prefix` a la ubicación dentro del bucket de Amazon S3 donde el trabajo de empaquetado almacena los artefactos de los componentes del modelo.
- (Opcional) `component_description` a una descripción para el componente del modelo.
- (Opcional) `tag_key1` y `tag_key2` a las claves de las etiquetas adjuntas al componente del modelo.
- (Opcional) `tag_value1` y `tag_value2` a los valores clave de las etiquetas adjuntas al componente del modelo.

```
aws lookoutvision start-model-packaging-job \
  --project-name project_name \
  --model-version model_version \
  --description="description" \
  --configuration
 "Greengrass={TargetPlatform={Os='LINUX',Arch='architecture',Accelerator='NVIDIA'},Compi
code\": \"gpu_code\", \"trt-ver\": \"trt_ver\", \"cuda-ver\":
 \"cuda_ver\"}',S3OutputLocation={Bucket='bucket',Prefix='prefix'},ComponentName='Compon
 {Key='tag_key2',Value='tag_value2'}]}" \
  --profile lookoutvision-access
```

Por ejemplo:

```
aws lookoutvision start-model-packaging-job \
  --project-name test-project-01 \
  --model-version 1 \
  --description="Model Packaging Job for G4 Instance using TargetPlatform
 Option" \
  --configuration
 "Greengrass={TargetPlatform={Os='LINUX',Arch='X86_64',Accelerator='NVIDIA'},CompilerOpt
code\": \"sm_75\", \"trt-ver\": \"7.1.3\", \"cuda-ver\":
 \"10.2\"}',S3OutputLocation={Bucket='bucket',Prefix='test-project-01/
folder'},ComponentName='SampleComponentNameX86TargetPlatform',ComponentVersion='0.1.0',C
 is my component',Tags=[{Key='modelKey0',Value='modelValue'},
 {Key='modelKey1',Value='modelValue'}]}" \
  --profile lookoutvision-access
```

Target Device

Utilice los siguientes comandos CLI para empaquetar un modelo para un dispositivo de destino.

Cambie los siguientes valores:

- `project_name` al nombre del proyecto que contiene el modelo que desea empaquetar.
- `model_version` a la versión del modelo que desea empaquetar.
- (Opcional) `description` a una descripción de su trabajo de empaquetado de modelos.
- `component_name`a un nombre para el componente del modelo en el que desea crear AWS IoT Greengrass V2.
- (Opcional) `component_version` a una versión del componente del modelo que crea el trabajo de empaquetado. Utilice el formato `major.minor.patch`. Por ejemplo, la 1.0.0 representa la primera versión principal de un componente.
- `bucket` al bucket de Amazon S3, donde el trabajo de empaquetado almacena los artefactos de los componentes del modelo.
- `prefix` a la ubicación dentro del bucket de Amazon S3 donde el trabajo de empaquetado almacena los artefactos de los componentes del modelo.
- (Opcional) `component_description` a una descripción para el componente del modelo.
- (Opcional) `tag_key1` y `tag_key2` a las claves de las etiquetas adjuntas al componente del modelo.
- (Opcional) `tag_value1` y `tag_value2` a los valores clave de las etiquetas adjuntas al componente del modelo.

```
aws lookoutvision start-model-packaging-job \
  --project-name project_name \
  --model-version model_version \
  --description="description" \
  --configuration
 "Greengrass={TargetDevice='jetson_xavier',S3OutputLocation={Bucket='bucket',Prefix='pre
 {Key='tag_key2',Value='tag_value2'}]}" \
  --profile lookoutvision-access
```

Por ejemplo:

```
aws lookoutvision start-model-packaging-job \
  --project-name project_01 \
  --model-version 1 \
```

```
 --description="description" \
 --configuration
"Greengrass={TargetDevice='jetson_xavier',S3OutputLocation={Bucket='bucket',Prefix='com
model component',Tags=[{Key='tag_key1',Value='tag_value1'},
{Key='tag_key2',Value='tag_value2'}]}" \
 --profile lookoutvision-access
```

4. Anote el valor de JobName en la respuesta. Lo necesitará en el siguiente paso. Por ejemplo:

```
{
    "JobName": "6bcfd0ff-90c3-4463-9a89-6b4be3daf972"
}
```

5. Utilice `DescribeModelPackagingJob` para obtener el estado actual del trabajo. Realice el siguiente cambio:

   - `project_name` al nombre del proyecto que está utilizando.
   - `job_name` al nombre del trabajo que anotó en el paso anterior.

```
aws lookoutvision describe-model-packaging-job \
    --project-name project_name \
    --job-name job_name \
    --profile lookoutvision-access
```

   El trabajo de empaquetado de modelos está completo si el valor de `Status` es `SUCCEEDED`. Si el valor es diferente, espere un minuto e inténtelo de nuevo.

6. Continúe con el despliegue utilizando AWS IoT Greengrass V2. Para obtener más información, consulte Implementación de sus componentes en un dispositivo.

Obtención de información sobre trabajos de empaquetado de modelos

Puede utilizar la consola AWS y el SDK de Amazon Lookout for Vision para obtener información sobre los trabajos de empaquetado de modelos que cree.

Temas

- Obtención de información sobre los trabajos de empaquetado de modelos (consola)
- Obtención de información sobre el trabajo de empaquetado de modelos (SDK)

Obtención de información sobre los trabajos de empaquetado de modelos (consola)

Para obtener información de trabajo de empaquetado de modelos (consola)

1. Abre la consola https://console.aws.amazon.com/lookoutvision/Amazon Lookout for Vision en.
2. Elija Comenzar.
3. En el panel de navegación izquierdo, elija Proyectos.
4. En la sección Proyectos, elija el proyecto que contiene el modelo de trabajo de empaquetado que desea ver.
5. En el panel de navegación izquierdo, en el nombre del proyecto, elija Paquetes de modelos de borde.
6. En la sección Trabajo de empaquetado de modelos, elija el trabajo de empaquetado de modelos que desee ver. Se muestra la página de detalles del trabajo de empaquetado de modelos.

Obtención de información sobre el trabajo de empaquetado de modelos (SDK)

Puede usar el AWS SDK para enumerar los trabajos de empaquetado de modelos de un proyecto y obtener información sobre un trabajo de empaquetado de modelos específico.

Enumere los trabajos de empaquetado de modelos

Puede enumerar los trabajos de empaquetado de modelos de un proyecto llamando a la ListModelPackagingJobsAPI. La respuesta incluye una lista de ModelPackagingJobMetadataobjetos que proporciona información sobre cada trabajo de empaquetado de modelos. También se incluye un token de paginación que puede usar para obtener el siguiente conjunto de resultados, si la lista está incompleta.

Para enumerar sus trabajos de empaquetado de modelos

1. Si aún no lo ha hecho, instale y configure el AWS CLI y el AWS SDKs. Para obtener más información, consulte Paso 4: Configura el AWS CLI y AWS SDKs.
2. Para ello, utilice el siguiente comando de la CLI: Cambie el valor de `project_name` por el nombre del proyecto que desee usar.

```
aws lookoutvision list-model-packaging-jobs \
```

```
  --project-name project_name \
  --profile lookoutvision-access
```

Describir un trabajo de empaquetado de modelos

Utilice la DescribeModelPackagingJobAPI para obtener información sobre un trabajo de empaquetado de modelos. La respuesta es un ModelPackagingDescriptionobjeto que incluye el estado actual del trabajo y otra información.

Para describir un paquete

1. Si aún no lo ha hecho, instale y configure el AWS CLI y el AWS SDKs. Para obtener más información, consulte Paso 4: Configura el AWS CLI y AWS SDKs.
2. Para ello, utilice el siguiente comando de la CLI: Realice el siguiente cambio:
   - `project_name` al nombre del proyecto que está utilizando.
   - `job_name` al nombre del trabajo. Recibirá el nombre del trabajo (`JobName`) cuando llame StartModelPackagingJob.

```
aws lookoutvision describe-model-packaging-job \
    --project-name project_name \
    --job-name job_name \
    --profile lookoutvision-access
```

Escribiendo el componente de la aplicación cliente

Un componente de aplicación cliente es un AWS IoT Greengrass Version 2 componente personalizado que usted escribe. Implementa la lógica empresarial necesaria para utilizar un modelo de Amazon Lookout for Vision en AWS IoT Greengrass Version 2 un dispositivo central.

Para acceder a un modelo, el componente de la aplicación cliente utiliza el componente Lookout for Vision Edge Agent. El componente Lookout for Vision Edge Agent proporciona una API que se utiliza para analizar imágenes con un modelo y administrar los modelos en un dispositivo principal.

La API de agente de Lookout for Vision Edge se implementa mediante gRPC, que es un protocolo para realizar llamadas a procedimientos remotos. Para obtener más información, consulte gRPC. Para escribir su código, puede usar cualquier lenguaje compatible con gRPC. Proporcionamos

un ejemplo de código Python. Para obtener más información, consulte Uso de un modelo en el componente de la aplicación cliente.

> **Note**
>
> El componente Lookout for Vision Edge Agent es una dependencia del componente del modelo que se implemente. Se implementa automáticamente en el dispositivo principal cuando se implementa el componente del modelo en el dispositivo principal.

Para escribir un componente de aplicación cliente, haga lo siguiente.

1. Configure su entorno para usar gRPC e instale bibliotecas de terceros.
2. Escribe código para usar el modelo.
3. Implemente el código como un componente personalizado en el dispositivo principal.

Para ver un ejemplo de un componente de aplicación cliente que muestra cómo realizar la detección de anomalías en una GStreamer canalización personalizada, consulte https://github.com/awslabs/aws-greengrass-labs-lookoutvision -gstreamer.

Configuración del entorno

Para escribir código de cliente, su entorno de desarrollo se conecta de forma remota a un dispositivo principal AWS IoT Greengrass Version 2 en el que ha implementado un componente y dependencias del modelo Amazon Lookout for Vision. Alternativamente, puede escribir código en un dispositivo principal. Para obtener más información, consulte Herramientas de desarrollo de AWS IoT Greengrass y Desarrollo de componentes de AWS IoT Greengrass.

Su código de cliente debe usar el cliente gRPC para acceder al agente de Amazon Lookout for Vision Edge. En esta sección se muestra cómo configurar su entorno de desarrollo con gRPC e instalar las dependencias de terceros necesarias para el código de ejemplo `DetectAnomalies`.

Cuando termine de escribir el código de su cliente, debe crear un componente personalizado e implementarlo en sus dispositivos de borde. Para obtener más información, consulte Creación del componente de la aplicación cliente.

Temas

- Configuración de gRPC

- Cómo añadir dependencias de terceros

Configuración de gRPC

En su entorno de desarrollo, necesita un cliente gRPC que utilice en su código para llamar a la API de agente de Lookout for Vision Edge. Para ello, debe crear un código auxiliar de gRPC mediante un archivo de definición de servicio `.proto` para el agente de Lookout for Vision Edge.

> **Note**
>
> También puede obtener el archivo de definición del servicio en el paquete de aplicaciones Lookout for Vision Edge Agent. El paquete de aplicaciones se instala cuando el componente Lookout for Vision Edge Agent se instala como una dependencia del componente del modelo. El paquete de aplicaciones se encuentra en `/greengrass/v2/packages/artifacts-unarchived/aws.iot.lookoutvision.EdgeAgent/`*`edge_agent_version`*`/lookoutvision_edge_agent`. Sustituya `edge_agent_version` por la versión del agente de Lookout for Vision Edge que esté utilizando. Para obtener el paquete de aplicaciones, debe implementar el agente Lookout for Vision Edge en un dispositivo principal.

Para configurar gRPC

1. Descarga el archivo zip, proto.zip. El archivo zip contiene el archivo de definición de servicio .proto (`edge-agent.proto`).
2. Descomprima el contenido.
3. Abra un símbolo del sistema y vaya a la carpeta que contiene `edge-agent.proto`.
4. Utilice los siguientes comandos para generar las interfaces de cliente de Python.

```
%%bash
python3 -m pip install grpcio
python3 -m pip install grpcio-tools
python3 -m grpc_tools.protoc --proto_path=. --python_out=. --grpc_python_out=.
 edge-agent.proto
```

Si los comandos se ejecutan correctamente, los stubs `edge_agent_pb2_grpc.py` y `edge_agent_pb2.py` se crean en el directorio de trabajo.

5. Escriba el código de cliente que usa su modelo. Para obtener más información, consulte Uso de un modelo en el componente de la aplicación cliente.

Cómo añadir dependencias de terceros

El código de ejemplo `DetectAnomalies` usa la biblioteca Pillow para trabajar con imágenes. Para obtener más información, consulte Detección de anomalías mediante bytes de imagen.

Utilice el siguiente comando para instalar la biblioteca de Pillow.

```
python3 -m pip install Pillow
```

Uso de un modelo en el componente de la aplicación cliente

Los pasos para usar un modelo de un componente de aplicación cliente son similares a los de usar un modelo alojado en la nube.

1. Comience a ejecutar el modelo.
2. Detecta anomalías en las imágenes.
3. Detenga el modelo si ya no se necesita.

El agente Amazon Lookout for Vision Edge proporciona una API para iniciar un modelo, detectar anomalías en una imagen y detener un modelo. También puede usar la API para enumerar los modelos de un dispositivo y obtener información sobre un modelo implementado. Para obtener más información, consulte Amazon Lookout for Vision Edge Agent API.

Puede obtener información sobre los errores comprobando los códigos de estado del gRPC. Para obtener más información, consulte Obtención de información de error.

Para escribir su código, puede usar cualquier lenguaje compatible con gRPC. Proporcionamos un ejemplo de código Python.

Temas

- Uso del código auxiliar en el componente de la aplicación cliente
- Iniciar el modelo
- Detección de anomalías
- Detener el modelo

- Mostrar modelos en un dispositivo
- Describir un modelo
- Obtención de información de error

Uso del código auxiliar en el componente de la aplicación cliente

Use el siguiente código para configurar el acceso a su modelo a través del agente de Lookout for Vision Edge.

```
import grpc
from edge_agent_pb2_grpc import EdgeAgentStub
import edge_agent_pb2 as pb2

# Creating stub.
 with grpc.insecure_channel("unix:///tmp/aws.iot.lookoutvision.EdgeAgent.sock") as
 channel:
    stub = EdgeAgentStub(channel)
    # Add additional code that works with Edge Agent in this block to prevent resources
 leakage
```

Iniciar el modelo

Para iniciar un modelo, se llama a la API StartModel. Es posible que el modelo tarde un poco en iniciarse. Puede comprobar el estado actual llamando a DescribeModel. El modelo se está ejecutando si el valor del campo `status` es En ejecución.

Código de ejemplo

*component_name*Sustitúyalo por el nombre del componente del modelo.

```
import time

import grpc
from edge_agent_pb2_grpc import EdgeAgentStub
import edge_agent_pb2 as pb2


model_component_name = "component_name"
```

```
def start_model_if_needed(stub, model_name):
    # Starting model if needed.
    while True:
        model_description_response =
 stub.DescribeModel(pb2.DescribeModelRequest(model_component=model_name))
        print(f"DescribeModel() returned {model_description_response}")
        if model_description_response.model_description.status == pb2.RUNNING:
            print("Model is already running.")
            break
        elif model_description_response.model_description.status == pb2.STOPPED:
            print("Starting the model.")
            stub.StartModel(pb2.StartModelRequest(model_component=model_name))
            continue
        elif model_description_response.model_description.status == pb2.FAILED:
            raise Exception(f"model {model_name} failed to start")
        print(f"Waiting for model to start.")
        if model_description_response.model_description.status != pb2.STARTING:
            break
        time.sleep(1.0)


# Creating stub.
with grpc.insecure_channel("unix:///tmp/aws.iot.lookoutvision.EdgeAgent.sock") as
 channel:
    stub = EdgeAgentStub(channel)
    start_model_if_needed(stub, model_component_name)
```

Detección de anomalías

La API DetectAnomalies se utiliza para detectar anomalías en una imagen.

La `DetectAnomalies` operación espera que el mapa de bits de la imagen se pase en formato RGB888 empaquetado. El primer byte representa el canal rojo, el segundo byte representa el canal verde y el tercer byte representa el canal azul. Si proporciona la imagen en un formato diferente, como BGR, las predicciones DetectAnomalies son incorrectas.

De forma predeterminada, OpenCV utiliza el formato BGR para los mapas de bits de las imágenes. Si utiliza OpenCV para capturar imágenes para `DetectAnomalies` analizarlas, debe convertir la imagen al formato antes de pasarla RGB888 a. `DetectAnomalies`

Las imágenes que suministre a `DetectAnomalies` deben tener las mismas dimensiones de ancho y alto que las imágenes que utilizó para entrenar el modelo.

Detección de anomalías mediante bytes de imagen

Puede detectar anomalías en una imagen proporcionando la imagen como bytes de imagen. En el siguiente ejemplo, los bytes de la imagen se recuperan de una imagen almacenada en el sistema de archivos local.

*sample.jpg*Sustitúyala por el nombre del archivo de imagen que desee analizar. *component_name*Sustitúyalo por el nombre del componente del modelo.

```
import time

from PIL import Image
import grpc
from edge_agent_pb2_grpc import EdgeAgentStub
import edge_agent_pb2 as pb2


model_component_name = "component_name"

....
# Detecting anomalies.
def detect_anomalies(stub, model_name, image_path):
    image = Image.open(image_path)
    image = image.convert("RGB")
    detect_anomalies_response = stub.DetectAnomalies(
        pb2.DetectAnomaliesRequest(
            model_component=model_name,
            bitmap=pb2.Bitmap(
                width=image.size[0],
                height=image.size[1],
                byte_data=bytes(image.tobytes())
            )
        )
    )
    print(f"Image is anomalous -
 {detect_anomalies_response.detect_anomaly_result.is_anomalous}")
    return detect_anomalies_response.detect_anomaly_result


# Creating stub.
with grpc.insecure_channel("unix:///tmp/aws.iot.lookoutvision.EdgeAgent.sock") as
 channel:
    stub = EdgeAgentStub(channel)
```

```
    start_model_if_needed(stub, model_component_name)
    detect_anomalies(stub, model_component_name, "sample.jpg")
```

Detección de anomalías mediante un segmento de memoria compartida

Puede detectar anomalías en una imagen proporcionando la imagen como bytes de imagen en el segmento de memoria compartida POSIX. Para obtener el mejor rendimiento, se recomienda utilizar memoria compartida para DetectAnomalies las solicitudes. Para obtener más información, consulte DetectAnomalies.

Detener el modelo

Si ya no utiliza el modelo, la API StopModel para detener la ejecución del modelo.

```
stop_model_response = stub.StopModel(
    pb2.StopModelRequest(
         model_component=model_component_name
    )
)
print(f"New status of the model is {stop_model_response.status}")
```

Mostrar modelos en un dispositivo

Puede usar la API the section called “ListModels” para enumerar los modelos que se implementan en un dispositivo.

```

models_list_response = stub.ListModels(
    pb2.ListModelsRequest()
)
for model in models_list_response.models:
    print(f"Model Details {model}")
```

Describir un modelo

Puede obtener información sobre un modelo que se implementa en un dispositivo llamando a la API DescribeModel. Usar `DescribeModel` es útil para obtener el estado actual de un modelo. Por ejemplo, necesita saber si un modelo está funcionando antes de poder llamar a `DetectAnomalies`. Para ver el código de ejemplo, consulte Iniciar el modelo.

Obtención de información de error

Los códigos de estado gRPC se utilizan para informar de los resultados de la API.

Puede obtener información de error detectando la excepción `RpcError`, tal como se muestra en el siguiente ejemplo. Para obtener información acerca de los códigos de error, consulte el tema de referencia de una API.

```
# Error handling.
try:
    stub.DetectAnomalies(detect_anomalies_request)
except grpc.RpcError as e:
    print(f"Error code: {e.code()}, Status: {e.details()}")
```

Creación del componente de la aplicación cliente

Puede crear el componente de la aplicación cliente una vez que haya generado los stubs de gRPC y tenga listo el código de la aplicación cliente. El componente que cree es un componente personalizado que se implementa en un dispositivo AWS IoT Greengrass Version 2 principal AWS IoT Greengrass V2. Una receta que cree describe su componente personalizado. La receta incluye todas las dependencias que también deban implementarse. En este caso, se especifica el componente de modelo que se crea en Empaquetado del modelo Amazon Lookout for Vision. Para obtener más información sobre estas recetas de componentes, consulte la AWS IoT Greengrass Version 2 Referencia de receta de componentes.

Los procedimientos de este tema muestran cómo crear el componente de la aplicación cliente a partir de un archivo de recetas y publicarlo como un componente AWS IoT Greengrass V2 personalizado. Puede usar la AWS IoT Greengrass V2 consola o el AWS SDK para publicar el componente.

Para obtener información detallada sobre la creación de un componente personalizado, consulte lo siguiente en la documentación de AWS IoT Greengrass V2.

- Desarrolle y pruebe un componente en su dispositivo
- Cree componentes AWS de IoT Greengrass
- Publique componentes para desplegarlos en sus dispositivos principales

Temas

- Permisos de IAM para publicar un componente de una aplicación cliente

- Creación de la receta
- Publicar el componente de la aplicación cliente (consola)
- Publicación del componente de aplicación cliente (SDK)

Permisos de IAM para publicar un componente de una aplicación cliente

Para crear y publicar el componente de la aplicación cliente, necesita los siguientes permisos de IAM:

- `greengrass:CreateComponentVersion`
- `greengrass:DescribeComponent`
- `s3:PutObject`

Creación de la receta

En este procedimiento, creará la receta para un componente de aplicación cliente sencillo. El código en `lookoutvision_edge_agent_example.py` muestra los modelos que se implementan en el dispositivo y se ejecutan automáticamente después de implementar el componente en el dispositivo principal. Para ver el resultado, compruebe el registro del componente después de implementarlo. Para obtener más información, consulte Implementación de sus componentes en un dispositivo. Cuando esté listo, utilice este procedimiento para crear la receta de código que implemente su lógica empresarial.

La receta se crea como un archivo con formato JSON o YAML. También cargue el código de la aplicación del cliente a un bucket de Amazon S3.

Para crear la receta del componente de la aplicación cliente

1. Si no lo ha hecho aún, debe crear los archivos de código auxiliar de gRPC. Para obtener más información, consulte Configuración de gRPC.
2. Guarde el siguiente código en un archivo denominado `lookoutvision_edge_agent_example.py`.

```
import grpc
from edge_agent_pb2_grpc import EdgeAgentStub
import edge_agent_pb2 as pb2

# Creating stub.
```

```
with grpc.insecure_channel("unix:///tmp/aws.iot.lookoutvision.EdgeAgent.sock") as
 channel:
    stub = EdgeAgentStub(channel)
    # Add additional code that works with Edge Agent in this block to prevent
 resources leakage

    models_list_response = stub.ListModels(
        pb2.ListModelsRequest()
    )
    for model in models_list_response.models:
        print(f"Model Details {model}")
```

3. Cree un bucket de Amazon S3 (o utilice uno existente) para almacenar los archivos fuente del componente de la aplicación cliente. El bucket debe estar en tu AWS cuenta y en la misma AWS región en la que utilizas AWS IoT Greengrass Version 2 Amazon Lookout for Vision.
4. Cargue `lookoutvision_edge_agent_example.py`, `edge_agent_pb2_grpc.py` and `edge_agent_pb2.py` en el bucket de Amazon S3 que creó en el paso anterior. Anote la ruta de Amazon S3 de cada archivo. Usted creó `edge_agent_pb2_grpc.py` y `edge_agent_pb2.py` en Configuración de gRPC.
5. En un editor, crea el siguiente archivo de recetas JSON o YAML.

   - `model_component`al nombre del componente del modelo. Para obtener más información, consulte Configuración de los componentes.
   - Cambie las entradas del URI por las rutas S3 de `lookoutvision_edge_agent_example.pyedge_agent_pb2_grpc.py`, `yedge_agent_pb2.py`.

   JSON

```
{
  "RecipeFormatVersion": "2020-01-25",
  "ComponentName": "com.lookoutvision.EdgeAgentPythonExample",
  "ComponentVersion": "1.0.0",
  "ComponentType": "aws.greengrass.generic",
  "ComponentDescription": "Lookout for Vision Edge Agent Sample Application",
  "ComponentPublisher": "Sample App Publisher",
  "ComponentDependencies": {
    "model_component": {
      "VersionRequirement": ">=1.0.0",
      "DependencyType": "HARD"
```

```
    }
  },
  "Manifests": [
    {
      "Platform": {
        "os": "linux"
      },
      "Lifecycle": {
        "install": "pip3 install grpcio grpcio-tools protobuf Pillow",
        "run": {
          "script": "python3 {artifacts:path}/
lookoutvision_edge_agent_example.py"
        }
      },
      "Artifacts": [
        {
          "Uri": "S3 path to lookoutvision_edge_agent_example.py"
        },
        {
          "Uri": "S3 path to edge_agent_pb2_grpc.py"
        },
        {
          "Uri": "S3 path to edge_agent_pb2.py"
        }
      ]
    }
  ],
  "Lifecycle": {}
}
```

YAML

```
---
RecipeFormatVersion: 2020-01-25
ComponentName: com.lookoutvison.EdgeAgentPythonExample
ComponentVersion: 1.0.0
ComponentDescription: Lookout for Vision Edge Agent Sample Application
ComponentPublisher: Sample App Publisher
ComponentDependencies:
  model_component:
    VersionRequirement: '>=1.0.0'
    DependencyType: HARD
Manifests:
```

```
- Platform:
    os: linux
  Lifecycle:
    install: |-
      pip3 install grpcio
      pip3 install grpcio-tools
      pip3 install protobuf
      pip3 install Pillow
    run:
      script: |-
        python3 {artifacts:path}/lookout_vision_agent_example.py
  Artifacts:
    - URI: S3 path to lookoutvision_edge_agent_example.py
    - URI: S3 path to edge_agent_pb2_grpc.py
    - URI: S3 path to edge_agent_pb2.py
```

6. Guarde el archivo JSON o YAML en su equipo.
7. Cree el componente de aplicación de cliente de la siguiente manera:

   - Si quieres usar la AWS IoT Greengrass consola, hazloPublicar el componente de la aplicación cliente (consola).
   - Si quieres usar el AWS SDK, hazloPublicación del componente de aplicación cliente (SDK).

Publicar el componente de la aplicación cliente (consola)

Puede usar la AWS IoT Greengrass V2 consola para publicar el componente de la aplicación cliente.

Para publicar el componente de la aplicación cliente

1. Si aún no lo ha hecho, cree la receta para el componente de la aplicación de su cliente de la siguiente manera. Creación de la receta
2. Abra la AWS IoT Greengrass consola en https://console.aws.amazon.com/iot/
3. En el panel de navegación izquierdo, en Greengrass, elija Componentes.
4. En Mis componentes, elija Crear componente.
5. En la página Crear componente, seleccione Introducir receta como JSON si desea utilizar una receta en formato JSON. Seleccione Introducir receta como YAML si quieres usar una receta en formato YAML.
6. En Receta, reemplace la receta existente por la receta JSON o YAML que creó en Creación de la receta.

7. Seleccione Crear componente.
8. A continuación, implemente el componente de la aplicación cliente.

Publicación del componente de aplicación cliente (SDK)

Puede publicar el componente de la aplicación cliente mediante la CreateComponentVersionAPI.

Para publicar el componente de aplicación cliente (SDK)

1. Si aún no lo ha hecho, cree la receta para el componente de la aplicación de su cliente de la siguiente manera. Creación de la receta
2. En el símbolo del sistema, escriba el siguiente comando para crear el componente de aplicación de cliente. Sustituya `recipe-file` por el nombre del archivo de recetas que creó en Creación de la receta.

   ```
   aws greengrassv2 create-component-version --inline-recipe fileb://recipe-file
   ```

   Anote el ARN del componente en la respuesta. Lo necesitará en el siguiente paso.
3. Utilice el siguiente comando para obtener el estado del componente de la aplicación de cliente. Sustituya `component-arn` por el ARN de la política anotado en el paso anterior. El componente de la aplicación cliente está listo si el valor de `componentState` es `DEPLOYABLE`.

   ```
   aws greengrassv2 describe-component --arn component-arn
   ```
4. A continuación, implemente el componente de la aplicación cliente.

Implementación de sus componentes en un dispositivo

Para implementar el componente del modelo y el componente de la aplicación cliente en un dispositivo AWS IoT Greengrass Version 2 principal, utilice la AWS IoT Greengrass V2 consola o la CreateDeploymentAPI. Para obtener más información, consulte Crear implementaciones en la Guía para desarrolladores de AWS IoT Greengrass Version 2 . Para obtener información sobre la actualización de un componente que se implementa en un dispositivo principal, consulte Revisar las implementaciones.

Temas

- Permisos de IAM para implementar componentes

- Implementación de sus componentes (consola)
- Implementación de los componentes (SDK)

Permisos de IAM para implementar componentes

Para implementar un componente AWS IoT Greengrass V2 , necesita los siguientes permisos:

- `greengrass:ListComponents`
- `greengrass:ListComponentVersions`
- `greengrass:ListCoreDevices`
- `greengrass:CreateDeployment`
- `greengrass:GetDeployment`
- `greengrass:ListDeployments`

`CreateDeployment`y `GetDeployment` tienen acciones dependientes. Para obtener más información, consulte Acciones definidas por AWS IoT Greengrass V2.

Para obtener más información acerca de la actualización de permisos en IAM, consulte Cambio de los permisos de un usuario de IAM.

Implementación de sus componentes (consola)

Utilice el siguiente procedimiento para implementar el componente de la aplicación cliente en un dispositivo principal. La aplicación cliente depende del componente del modelo (que a su vez depende del agente Lookout for Vision Edge). Al implementar el componente de aplicación cliente, también se inicia la implementación del componente modelo y del agente Lookout for Vision Edge.

> **Note**
>
> Puede añadir sus componentes a una implementación existente. También puede implementar componentes en un grupo de cosas.

Para ejecutar este procedimiento, debe tener un dispositivo AWS IoT Greengrass V2 principal configurado. Para obtener más información, consulte Configuración del dispositivo AWS IoT Greengrass Version 2 principal.

Para implementar sus componentes en un dispositivo

1. Abra la AWS IoT Greengrass consola en https://console.aws.amazon.com/iot/.
2. En el panel de navegación izquierdo, en Greengrass, elija Implementaciones.
3. En Implementaciones, elija Crear.
4. En la página Especificar detalles, haga lo siguiente:
    1. En Información de la implementación, introduzca o modifique el nombre descriptivo de su implementación.
    2. En Destino de implementación, seleccione Dispositivo principal e introduzca un nombre de destino.
    3. Elija Next (Siguiente).
5. En la página Seleccionar componentes, haga lo siguiente:
    1. En Mis componentes, elija el nombre del componente de la aplicación cliente (`com.lookoutvison.EdgeAgentPythonExample`).
    2. Elija Siguiente.
6. En la página Configurar componentes, mantenga la configuración actual y seleccione Siguiente.
7. En la página Configurar los ajustes avanzados, conserve los ajustes actuales y seleccione Siguiente.
8. En la página de revisión, elija Implementar para empezar a implementar el componente.

Comprobar el estado de la implementación (consola)

Puede comprobar el estado de la implementación desde la AWS IoT Greengrass V2 consola. Si el componente de la aplicación cliente utiliza la receta y el código del ejemplothe section called “Creación del componente de la aplicación cliente”, consulte el registro del componente de la aplicación cliente una vez finalizada la implementación. Si se ejecuta correctamente, el registro incluye una lista de los modelos de Lookout for Vision que se han desplegado en el componente.

Para obtener información sobre el uso del AWS SDK para comprobar el estado de la implementación, consulte Comprobar el estado de la implementación.

Para comprobar el estado de la implementación

1. Abre la AWS IoT Greengrass consola en https://console.aws.amazon.com/iot/

2. En el panel de navegación de la izquierda, elija Dispositivos principales.
3. En los dispositivos principales de Greengrass, seleccione su dispositivo principal.
4. Seleccione la pestaña Implementaciones para ver el estado actual de la implementación.
5. Cuando las implementaciones se hayan realizado correctamente (el estado es Completado), abra una ventana de terminal en el dispositivo principal y consulte el registro de los componentes de la aplicación cliente en `/greengrass/v2/logs/com.lookoutvison.EdgeAgentPythonExample.log`. Si la implementación utiliza la receta y el código de ejemplo, el registro incluye el resultado de `lookoutvision_edge_agent_example.py`. Por ejemplo:

```
Model Details model_component:"ModelComponent"
```

Implementación de los componentes (SDK)

Utilice el siguiente procedimiento para implementar el componente de la aplicación cliente, el componente del modelo y el agente Amazon Lookout for Vision Edge en su dispositivo principal.

1. Cree un `deployment.json` archivo para definir la configuración de implementación de sus componentes. Este archivo debería ser igual al siguiente ejemplo.

```
{
  "targetArn":"targetArn",
  "components": {
    "com.lookoutvison.EdgeAgentPythonExample": {
      "componentVersion": "1.0.0",
      "configurationUpdate": {
      }
    }
  }
}
```

   - En el campo `targetArn`, sustituya *`targetArn`* por el nombre de recurso de Amazon (ARN) de la cosa o grupo de cosas a la que apunte la implementación, en el siguiente formato:
     - Cosa: `arn:aws:iot:`*`region`*`:`*`account-id`*`:thing/`*`thingName`*
     - Grupo de cosas: `arn:aws:iot:`*`region`*`:`*`account-id:`*`thinggroup/`*`thingGroupName`*
2. Compruebe si el objetivo de la implementación tiene una implementación existente que desee revisar. Haga lo siguiente:

a. Ejecute el siguiente comando para enumerar las implementaciones del destino de implementación. `targetArn`Sustitúyalo por el nombre de recurso de Amazon (ARN) de la cosa o grupo de cosas de AWS IoT objetivo. Para obtener todos los elementos ARNs de la región de AWS actual, utilice el comando`aws iot list-things`.

```
aws greengrassv2 list-deployments --target-arn targetArn
```

La respuesta contiene una lista con la implementación más reciente del objetivo. Si la respuesta está vacía, significa que el destino no tiene una implementación existente y puede ir directamente al paso 3. De lo contrario, copie el `deploymentId` de la respuesta para usarlo en el paso siguiente.

b. Ejecute el siguiente comando para obtener los detalles de la implementación. Estos detalles incluyen los metadatos, los componentes y la configuración del trabajo. Sustituya `deploymentId` por el ID de la tarea del paso anterior.

```
aws greengrassv2 get-deployment --deployment-id deploymentId
```

c. Copia cualquiera de los siguientes pares clave-valor de la respuesta del comando anterior a deployment.json. Puede cambiar estos valores para la nueva implementación.

- `deploymentName`: el nombre de la implementación.
- `components`: los componentes de la implementación. Para desinstalar un componente, elimínelo de este objeto.
- `deploymentPolicies`: las políticas de la implementación.
- `tags`: las etiquetas de la implementación.

3. Ejecute el siguiente comando para implementar los componentes en el dispositivo. Anote el valor de `deploymentId` en la respuesta.

```
aws greengrassv2 create-deployment \
    --cli-input-json file://path/to/deployment.json
```

4. Ejecute el siguiente comando para verificar el estado de la implementación. Cambie `deployment-id` al valor que anotó en el paso anterior. La implementación se ha completado correctamente si el valor de `deploymentStatus` es `COMPLETED`.

```
aws greengrassv2 get-deployment --deployment-id deployment-id
```

5. Cuando las implementaciones se realicen correctamente, abra una ventana de terminal en el dispositivo principal y consulte el registro de componentes de la aplicación cliente en `/greengrass/v2/logs/com.lookoutvison.EdgeAgentPythonExample.log`. Si la implementación utiliza la receta y el código de ejemplo, el registro incluye el resultado de `lookoutvision_edge_agent_example.py`. Por ejemplo:

```
Model Details model_component:"ModelComponent"
```

Amazon Lookout for Vision Edge Agent API

Esta sección es la referencia de la API para el agente de Amazon Lookout for Vision Edge.

Detección de anomalías con un modelo

La DetectAnomalies API se utiliza para detectar anomalías en las imágenes mediante un modelo en ejecución en un dispositivo AWS IoT Greengrass Version 2 principal.

Obtención de información del modelo

APIs que obtienen información sobre los modelos implementados en un dispositivo principal.

- ListModels
- DescribeModel

Ejecutar un modelo

APIs para iniciar y detener un modelo de Amazon Lookout for Vision desplegado en un dispositivo central.

- StartModel
- StopModel

DetectAnomalies

Detecta anomalías en la imagen suministrada.

La respuesta de `DetectAnomalies` incluye una predicción booleana de que la imagen contiene una o más anomalías y un valor de confianza para la predicción. Si el modelo es un modelo de segmentación, la respuesta incluye lo siguiente:

- Una imagen de máscara que cubre cada tipo de anomalía con un color único. Puede hacer que `DetectAnomalies` guarde la imagen de la máscara en la memoria compartida o la devuelva en forma de bytes de la imagen.
- El porcentaje de área de la imagen que cubre un tipo de anomalía.
- El color hexadecimal de un tipo de anomalía en la imagen de la máscara.

> **Note**
>
> El modelo que utiliza con `DetectAnomalies` debe estar funcionando. Puede volver a visualizar el estado actual llamando a DescribeModel. Para iniciar un modelo en ejecución, consulte StartModel.

`DetectAnomalies`admite mapas de bits (imágenes) empaquetados en formato RGB888 intercalado. El primer byte representa el canal rojo, el segundo byte representa el canal verde y el tercer byte representa el canal azul. Si proporciona la imagen en un formato diferente, como BGR, las predicciones son incorrectas. DetectAnomalies

De forma predeterminada, OpenCV utiliza el formato BGR para los mapas de bits de las imágenes. Si utiliza OpenCV para capturar imágenes para `DetectAnomalies` analizarlas, debe convertir la imagen al formato antes de pasarla RGB888 a. `DetectAnomalies`

La dimensión mínima de imagen admitida es de 64 x 64 píxeles. La dimensión máxima de imagen admitida es de 4096 x 4096 píxeles.

Puede enviar la imagen en el mensaje protobuf o a través de un segmento de memoria compartida. La serialización de imágenes de gran tamaño en el mensaje protobuf puede aumentar considerablemente la latencia de las llamadas a `DetectAnomalies`. Para reducir al mínimo la latencia, le recomendamos que utilice memoria compartida.

```
rpc DetectAnomalies(DetectAnomaliesRequest) returns (DetectAnomaliesResponse);
```

DetectAnomaliesRequest

Los parámetros de entrada de `DetectAnomalies`.

```
message Bitmap {
  int32 width = 1;
  int32 height = 2;
  oneof data {
    bytes byte_data = 3;
    SharedMemoryHandle shared_memory_handle = 4;
  }
}
```

```
message SharedMemoryHandle {
  string name = 1;
  uint64 size = 2;
  uint64 offset = 3;
}
```

```
message AnomalyMaskParams {
SharedMemoryHandle shared_memory_handle = 2;
}
```

```
message DetectAnomaliesRequest {
string model_component = 1;
Bitmap bitmap = 2;
AnomalyMaskParams anomaly_mask_params = 3;
}
```

BITMAP

La imagen que desea analizar con `DetectAnomalies`.

width

El ancho de la imagen en píxeles.

height

La altura de la imagen en píxeles.

byte_data

Los bytes de imagen se transmiten en el mensaje protobuf.

shared_memory_handle

Los bytes de imagen se transfieren al segmento de memoria compartida.

SharedMemoryHandle

Representa un segmento de memoria compartida POSIX.

nombre

Nombre del segmento de memoria POSIX. Para obtener información sobre la creación de memoria compartida, consulte shm_open.

tamaño

El tamaño del búfer de la imagen en bytes a partir del desplazamiento.

desplazamiento

El desplazamiento, en bytes, al principio del búfer de imagen desde el inicio del segmento de memoria compartida.

AnomalyMaskParams

Parámetros para generar una máscara de anomalías. (Modelo de segmentación).

shared_memory_handle

Contiene los bytes de imagen de la máscara, si no se especificó `shared_memory_handle`.

DetectAnomaliesRequest

model_component

El nombre del AWS IoT Greengrass V2 componente que contiene el modelo que desea utilizar.

BITMAP

La imagen que desea analizar con `DetectAnomalies`.

anomaly_mask_params

Parámetros opcionales para generar la máscara. (Modelo de segmentación).

DetectAnomaliesResponse

La respuesta de `DetectAnomalies`.

```
message DetectAnomalyResult {
bool is_anomalous = 1;
float confidence = 2;
Bitmap anomaly_mask = 3;
repeated Anomaly anomalies = 4;
float anomaly_score = 5;
float anomaly_threshold = 6;
}
```

```
message Anomaly {
string name = 1;
PixelAnomaly pixel_anomaly = 2;
```

```
message PixelAnomaly {
float total_percentage_area = 1;
string hex_color = 2;
}
```

```
message DetectAnomaliesResponse {
  DetectAnomalyResult detect_anomaly_result = 1;
}
```

Anomalía

Representa una anomalía encontrada en una imagen. (Modelo de segmentación).

nombre

El nombre de un tipo de anomalía que se encuentra en una imagen. `name`ese asigna a un tipo de anomalía del conjunto de datos de entrenamiento. El servicio inserta automáticamente el tipo de anomalía de fondo en la respuesta desde DetectAnomalies.

pixel_anomaly

Información sobre la máscara de píxeles que cubre un tipo de anomalía.

PixelAnomaly

Información sobre la máscara de píxeles que cubre un tipo de anomalía. (Modelo de segmentación).

total_percentage_area

El porcentaje de área de la imagen que cubre el tipo de anomalía.

hex_color

Un valor de color hexadecimal que representa el tipo de anomalía de la imagen. El color se corresponde con el color del tipo de anomalía utilizado en el conjunto de datos de entrenamiento.

DetectAnomalyResult

is_anomalous

Indica si la imagen contiene una anomalía. `true` si la imagen contiene una anomalía. `false` si la imagen es normal.

confidence

La confianza que `DetectAnomalies` se deposita en la precisión de la predicción. `confidencees` un valor de punto flotante entre 0 y 1.

anomaly_mask

si no se proporcionó shared_memory_handle, contiene los bytes de imagen de la máscara. (Modelo de segmentación).

Anomalías

Una lista de 0 o más anomalías encontradas en la imagen de entrada. (Modelo de segmentación).

anomaly_score

Número que cuantifica en qué medida las anomalías pronosticadas para una imagen se desvían de una imagen sin anomalías. `anomaly_score` es un valor flotante que va desde `0.0` hasta

(desviación más baja con respecto a una imagen normal) 1,0 (desviación más alta con respecto a una imagen normal). Amazon Lookout for Vision devuelve un valor para `anomaly_score`, incluso si la predicción de una imagen es normal.

anomaly_threshold

Número (flotante) que determina si la clasificación prevista para una imagen es normal o anómala. Las imágenes con una `anomaly_score` igual o superior al valor de `anomaly_threshold` se consideran anómalas. Un valor de `anomaly_score` inferior a `anomaly_threshold` indica una imagen normal. Amazon Lookout for Vision calcula el valor de `anomaly_threshold` que utiliza un modelo al entrenar el modelo. No puede establecer ni cambiar el valor de `anomaly_threshold`.

Códigos de estado

Código	Número	Descripción
OK (Correcto)	0	`DetectAnomalies` realizó una predicción correctamente
UNKNOWN	2	Se ha producido un error desconocido.
INVALID_ARGUMENT	3	Uno o varios parámetros de entrada no son válidos. Consulte el mensaje de error para obtener más información.
NOT_FOUND	5	No se encontró un modelo con el nombre especificado.
RESOURCE_EXHAUSTED	8	No hay recursos suficientes para realizar esta operación . Por ejemplo, The Lookout for Vision Edge Agent no puede mantener el ritmo de las llamadas a `DetectAnomalies` . Consulte el mensaje de error para obtener más información.

Código	Número	Descripción
FAILED_PRECONDITION	9	`DetectAnomalies` se solicitó para un modelo que no está en estado EN EJECUCIÓN.
INTERNAL	13	Se ha producido un error de servicio interno.

DescribeModel

Describe un modelo de Amazon Lookout for Vision que se implementa en AWS IoT Greengrass Version 2 un dispositivo principal.

```
rpc DescribeModel(DescribeModelRequest) returns (DescribeModelResponse);
```

DescribeModelRequest

```
message DescribeModelRequest {
  string model_component = 1;
}
```

model_component

El nombre del AWS IoT Greengrass V2 componente que contiene el modelo que desea describir.

DescribeModelResponse

```
message ModelDescription {
  string model_component = 1;
  string lookout_vision_model_arn = 2;
  ModelStatus status = 3;
  string status_message = 4;
}
```

```
message DescribeModelResponse {
  ModelDescription model_description = 1;
}
```

ModelDescription

model_component

El nombre del AWS IoT Greengrass Version 2 componente que contiene el modelo Amazon Lookout for Vision.

lookout_vision_model_arn

El nombre del recurso de Amazon (ARN) del modelo Amazon Lookout for Vision que se utilizó para generar el componente. AWS IoT Greengrass V2

status

El estado actual del modelo. Para obtener más información, consulte ModelStatus.

status_message

El mensaje de estado del modelo.

Códigos de estado

Código	Número	Descripción
OK (Correcto)	0	La llamada se ha realizado correctamente.
UNKNOWN	2	Se ha producido un error desconocido.
INVALID_ARGUMENT	3	Uno o varios parámetros de entrada no son válidos. Consulte el mensaje de error para obtener más información.
NOT_FOUND	5	No se encontró un modelo con el nombre indicado.

Código	Número	Descripción
INTERNAL	13	Se ha producido un error de servicio interno.

ListModels

Muestra los modelos implementados en un dispositivo AWS IoT Greengrass Version 2 principal.

```
rpc ListModels(ListModelsRequest) returns (ListModelsResponse);
```

ListModelsRequest

```
message ListModelsRequest {}
```

ListModelsResponse

```
message ModelMetadata {
  string model_component = 1;
  string lookout_vision_model_arn = 2;
  ModelStatus status = 3;
  string status_message = 4;
}
```

```
message ListModelsResponse {
  repeated ModelMetadata models = 1;
}
```

ModelMetadata

model_component

El nombre del AWS IoT Greengrass Version 2 componente que contiene un modelo de Amazon Lookout for Vision.

lookout_vision_model_arn

El nombre del recurso de Amazon (ARN) del modelo Amazon Lookout for Vision que se utilizó para generar el componente. AWS IoT Greengrass V2

status

El estado actual del modelo. Para obtener más información, consulte ModelStatus.

status_message

El mensaje de estado del modelo.

Códigos de estado

Código	Número	Descripción
OK (Correcto)	0	La llamada se ha realizado correctamente.
UNKNOWN	2	Se ha producido un error desconocido.
INTERNAL	13	Se ha producido un error de servicio interno.

StartModel

Inicia un modelo que se ejecuta en un dispositivo AWS IoT Greengrass Version 2 central. Es posible que el modelo tarde un poco en empezar a funcionar. Para comprobar el estado actual, llame a DescribeModel. El modelo se está ejecutando si el `Status` campo lo está`RUNNING`.

La cantidad de modelos que puede ejecutar simultáneamente depende de la especificación de hardware del dispositivo principal.

```
rpc StartModel(StartModelRequest) returns (StartModelResponse);
```

StartModelRequest

```
message StartModelRequest {
  string model_component = 1;
}
```

model_component

El nombre del AWS IoT Greengrass Version 2 componente que contiene el modelo que desea iniciar.

StartModelResponse

```
message StartModelResponse {
  ModelStatus status = 1;
}
```

status

El estado actual del modelo. La respuesta es `STARTING` si la llamada se realiza correctamente. Para obtener más información, consulte ModelStatus.

Códigos de estado

Código	Número	Descripción
OK (Correcto)	0	El modelo se está iniciando
UNKNOWN	2	Se ha producido un error desconocido.
INVALID_ARGUMENT	3	Uno o varios parámetros de entrada no son válidos. Consulte el mensaje de error para obtener más información.
NOT_FOUND	5	No se encontró un modelo con el nombre indicado.
RESOURCE_EXHAUSTED	8	No hay recursos suficientes para realizar esta operación. Por ejemplo, no hay memoria suficiente para cargar el

Código	Número	Descripción
		modelo. Consulte el mensaje de error para obtener más información.
FAILED_PRECONDITION	9	Se utilizó el método para un modelo que no se encuentra en el estado DETENIDO o FALLIDO.
INTERNAL	13	Se ha producido un error de servicio interno.

StopModel

Detiene la ejecución de un modelo en un dispositivo AWS IoT Greengrass Version 2 central. `StopModel`regresa después de que el modelo se haya detenido. El modelo se ha detenido correctamente si el campo `Status` de la respuesta es `STOPPED`.

```
  rpc StopModel(StopModelRequest) returns (StopModelResponse);
```

StopModelRequest

```
message StopModelRequest {
  string model_component = 1;
}
```

model_component

El nombre del AWS IoT Greengrass Version 2 componente que contiene el modelo que desea detener.

StopModelResponse

```
message StopModelResponse {
  ModelStatus status = 1;
```

```
}
```

status

El estado actual del modelo. La respuesta es `STOPPED` si la llamada se realiza correctamente. Para obtener más información, consulte ModelStatus.

Códigos de estado

Código	Número	Descripción
OK (Correcto)	0	El modelo se detiene.
UNKNOWN	2	Se ha producido un error desconocido.
INVALID_ARGUMENT	3	Uno o varios parámetros de entrada no son válidos. Consulte el mensaje de error para obtener más información.
NOT_FOUND	5	No se encontró un modelo con el nombre indicado.
FAILED_PRECONDITION	9	Se utilizó el método para un modelo que no está en estado EN EJECUCIÓN.
INTERNAL	13	Se ha producido un error de servicio interno.

ModelStatus

El estado de un modelo que se implementa en un dispositivo AWS IoT Greengrass Version 2 principal. Para comprobar el estado actual, llame a DescribeModel.

```
enum ModelStatus {
  STOPPED = 0;
  STARTING = 1;
```

```
  RUNNING = 2;
  FAILED = 3;
  STOPPING = 4;
}
```

Uso del panel de Amazon Lookout for Vision

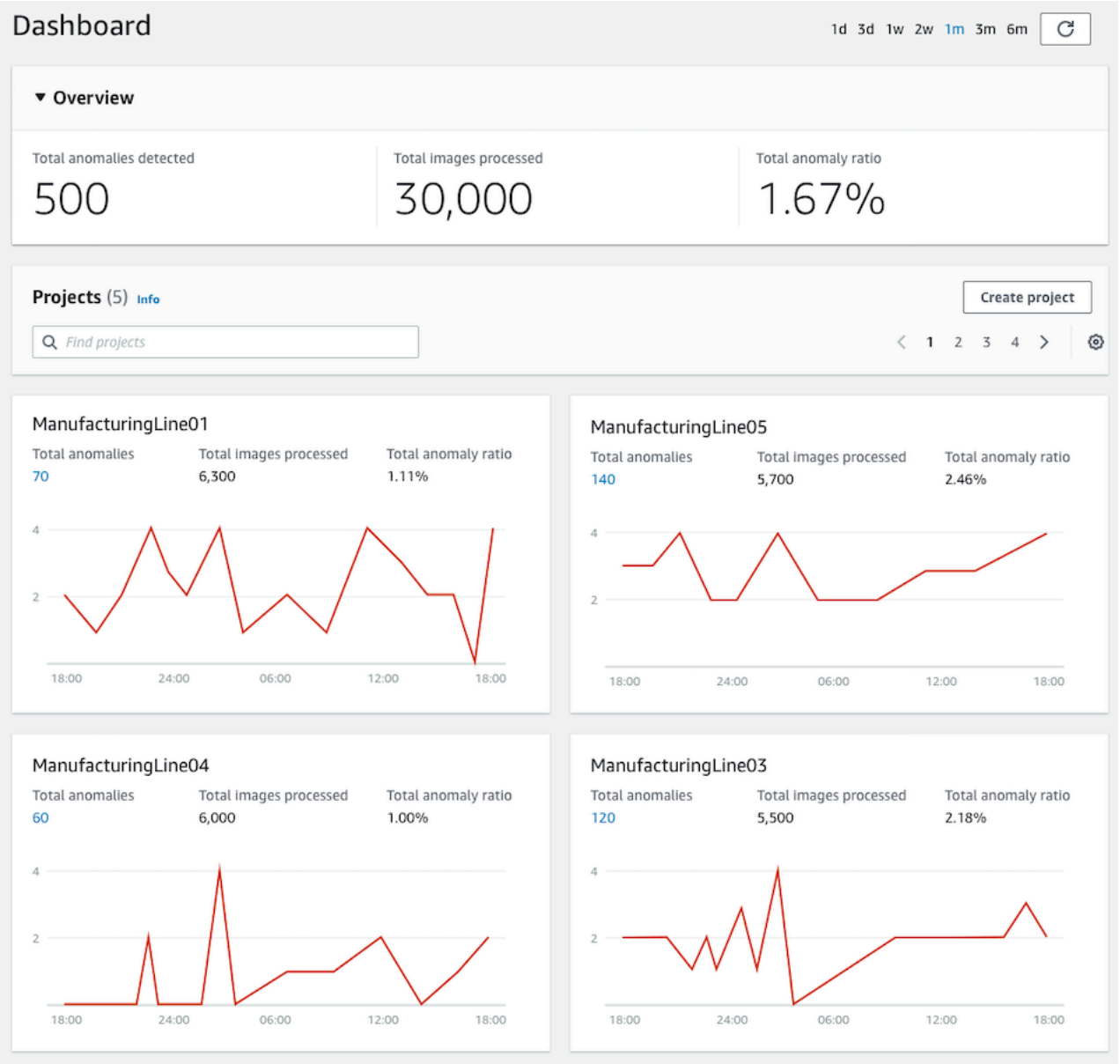
El panel proporciona una visión general de las métricas de sus proyectos de Amazon Lookout for Vision, como el número total de anomalías detectadas durante la última semana. Con el panel, obtiene una visión general de todos sus proyectos y una visión general de cada proyecto individual. Puede elegir el cronograma en el que se muestran las métricas. También puede utilizar el panel para crear un nuevo proyecto.

La sección Descripción general muestra el número total de proyectos, el número total de imágenes y el número total de imágenes detectadas en todos sus proyectos.

La sección Proyectos muestra la siguiente información general para proyectos individuales:

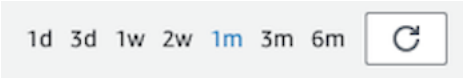
- El número total de anomalías detectadas.
- El número total de imágenes procesadas.
- La proporción total de anomalías (es decir, el porcentaje de imágenes detectadas con una anomalía).
- Un gráfico muestra las detecciones de anomalías durante el período de tiempo elegido.

También puede obtener más información sobre un proyecto.

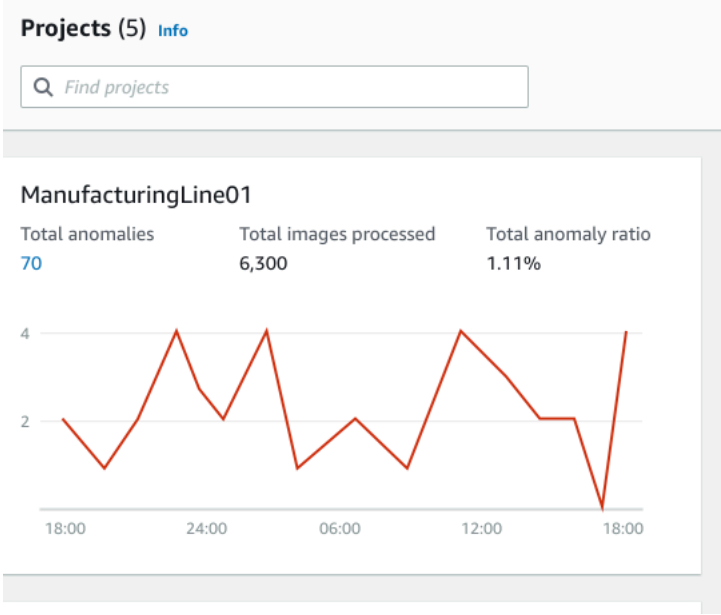


Cómo usar el panel

1. Abre la consola https://console.aws.amazon.com/lookoutvision/Amazon Lookout for Vision en.
2. Elija Comenzar.
3. En el panel de navegación izquierdo, elija Panel.
4. Para consultar las métricas durante un período de tiempo específico, haga lo siguiente:
   a. Elija el período de tiempo en la parte superior derecha del panel de control.
   b. Elija el botón de actualización para mostrar el panel de control con la nueva línea temporal.



5. Para obtener más detalles sobre un proyecto, elija el nombre del proyecto en la sección Proyectos (por ejemplo, `ManufacturingLine01`).



6. Para crear un proyecto, seleccione Crear proyecto en la sección Proyectos.

Administrar los recursos de Amazon Lookout for Vision

Puede administrar sus recursos de Amazon Lookout for Vision mediante la consola o AWS el SDK. Amazon Lookout for Vision tiene los siguientes recursos:

- Proyectos
- Conjuntos de datos
- Modelos
- Pruebas de detección

> **Note**
>
> No puede eliminar una tarea de pruebas de detección. Además, no puede administrar las detecciones de prueba mediante el AWS SDK.

Temas

Visualización de sus proyectos

Puede obtener una lista de los proyectos de Amazon Lookout for Vision e información sobre proyectos individuales desde la consola o mediante AWS el SDK.

> **Note**
>
> Al final, la lista de proyectos es coherente. Si crea o elimina un proyecto, puede que tenga que esperar un poco antes de que la lista de proyectos esté actualizada.

Ver sus proyectos (consola)

Realice los pasos que se indican en el procedimiento siguiente para detener el modelo usando la consola.

Para ver los proyectos

1. Abre la consola https://console.aws.amazon.com/lookoutvision/Amazon Lookout for Vision en.
2. Elija Comenzar.
3. En el panel de navegación izquierdo, elija Proyectos. Se abrirá la vista de proyectos.
4. Elija un nombre de proyecto para ver los detalles del proyecto.

Visualización de sus proyectos (SDK)

Un proyecto administra los conjuntos de datos y los modelos para un único caso de uso. Por ejemplo, detectar anomalías en piezas de máquinas. En el siguiente ejemplo, se llama a `ListProjects` para ver una lista de sus proyectos.

Para ver los proyectos (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: Configura el AWS CLI y AWS SDKs.
2. Utilice el siguiente código de ejemplo para ver sus proyectos.

   CLI

   Use el comando `list-projects` para ver los proyectos en su cuenta.

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

Para obtener más información sobre un proyecto, use el comando `describe-project`.

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

```
aws lookoutvision describe-project --project-name project_name \
  --profile lookoutvision-access
```

Python

Este código se ha tomado del GitHub repositorio de ejemplos del SDK de AWS documentación. Consulte el ejemplo completo aquí.

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

        :param lookoutvision_client: A Boto3 Lookout for Vision client.
        """
        try:
            response = lookoutvision_client.list_projects()
            for project in response["Projects"]:
                print("Project: " + project["ProjectName"])
                print("\tARN: " + project["ProjectArn"])
                print("\tCreated: " + str(["CreationTimestamp"]))
                print("Datasets")
                project_description = lookoutvision_client.describe_project(
                    ProjectName=project["ProjectName"]
                )
                if not project_description["ProjectDescription"]["Datasets"]:
                    print("\tNo datasets")
                else:
                    for dataset in project_description["ProjectDescription"][
                        "Datasets"
                    ]:
                        print(f"\ttype: {dataset['DatasetType']}")
                        print(f"\tStatus: {dataset['StatusMessage']}")

                print("Models")
                response_models = lookoutvision_client.list_models(
                    ProjectName=project["ProjectName"]
```

```
                )
                if not response_models["Models"]:
                    print("\tNo models")
                else:
                    for model in response_models["Models"]:
                        Models.describe_model(
                            lookoutvision_client,
                            project["ProjectName"],
                            model["ModelVersion"],
                        )


 print("-----------------------------------------------------------\n")
            print("Done!")
        except ClientError:
            logger.exception("Problem listing projects.")
            raise
```

Java V2

Este código se ha tomado del GitHub repositorio de ejemplos del SDK de AWS documentación. Consulte el ejemplo completo aquí.

```
/**
 * Lists the Amazon Lookout for Vision projects in the current AWS account and
 AWS
 * Region.
 *
 * @param lfvClient   An Amazon Lookout for Vision client.
 * @param projectName The name of the project that you want to create.
 * @return List<ProjectMetadata> Metadata for each project.
 */
public static List<ProjectMetadata> listProjects(LookoutVisionClient lfvClient)
                throws LookoutVisionException {

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

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

```
        ListProjectsIterable projects =
 lfvClient.listProjectsPaginator(listProjectsRequest);

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

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

        return projectMetadata;

}
```

Eliminación de un proyecto

Puede eliminar un proyecto desde la página de visualización de proyectos de la consola o mediante la operación `DeleteProject`.

Las imágenes a las que hacen referencia los conjuntos de datos de un proyecto no se eliminan.

Eliminación de un proyecto (consola)

Utilice el siguiente procedimiento para eliminar un proyecto. Si utiliza el procedimiento de consola, las versiones del modelo y los conjuntos de datos asociados se eliminarán automáticamente.

Para eliminar un proyecto

1. Abre la consola https://console.aws.amazon.com/lookoutvision/Amazon Lookout for Vision en.
2. Elija Comenzar.
3. En el panel de navegación izquierdo, elija Proyectos.
4. En la página Proyectos, elija el proyecto que desea eliminar.
5. En la parte superior de la página, elija Eliminar.
6. En el cuadro de diálogo Eliminar, escriba eliminar para confirmar que desea eliminar el proyecto.
7. Si es necesario, elija eliminar todos los conjuntos de datos y modelos asociados.
8. Elija Delete project (Eliminar proyecto).

Eliminación de un proyecto (SDK)

Para eliminar un proyecto de Amazon Lookout for Vision, debe DeleteProjectllamar y proporcionar el nombre del proyecto que desea eliminar.

Antes de eliminar un proyecto, primero debe eliminar todos los modelos del proyecto. Para obtener más información, consulte Eliminación de un modelo (SDK). También debe eliminar los conjuntos de datos asociados al modelo. Para obtener más información, consulte Eliminación de un conjunto 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 `DeleteProject` 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: Configura el AWS CLI y AWS SDKs.
2. Utilice el código siguiente para eliminar un proyecto.

   AWS CLI

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

   ```
   aws lookoutvision delete-project --project-name project_name \
     --profile lookoutvision-access
   ```

   Python

   Este código se ha tomado del GitHub repositorio de ejemplos del SDK de AWS documentación. Consulte el ejemplo completo aquí.

   ```
       @staticmethod
       def delete_project(lookoutvision_client, project_name):
           """
           Deletes a Lookout for Vision Model

           :param lookoutvision_client: A Boto3 Lookout for Vision client.
           :param project_name: The name of the project that you want to delete.
           """
   ```

```
        try:
            logger.info("Deleting project: %s", project_name)
            response =
 lookoutvision_client.delete_project(ProjectName=project_name)
            logger.info("Deleted project ARN: %s ", response["ProjectArn"])
        except ClientError as err:
            logger.exception("Couldn't delete project %s.", project_name)
            raise
```

Java V2

Este código se ha tomado del GitHub repositorio de ejemplos del SDK de AWS documentación. Consulte el ejemplo completo aquí.

```
/**
 * Deletes an Amazon Lookout for Vision project.
 *
 * @param lfvClient   An Amazon Lookout for Vision client.
 * @param projectName The name of the project that you want to create.
 * @return String The ARN of the deleted project.
 */
public static String deleteProject(LookoutVisionClient lfvClient, String
 projectName)
                throws LookoutVisionException {

        logger.log(Level.INFO, "Deleting project: {0}", projectName);

        DeleteProjectRequest deleteProjectRequest =
 DeleteProjectRequest.builder()
                        .projectName(projectName)
                        .build();

        DeleteProjectResponse response =
 lfvClient.deleteProject(deleteProjectRequest);

        logger.log(Level.INFO, "Deleted project: {0} ARN: {1}",
                        new Object[] { projectName, response.projectArn() });

        return response.projectArn();

}
```

Visualización de sus conjuntos de datos

Un proyecto puede tener un único conjunto de datos que se utilice para entrenar y probar el modelo. Como alternativa, puede tener conjuntos de datos de entrenamiento y prueba independientes. Puede utilizar la consola para ver los conjuntos de datos. También puede usar la `DescribeDataset` operación para obtener información sobre un conjunto de datos (entrenamiento o prueba).

Ver los conjuntos de datos de un proyecto (consola)

Realice los pasos que se indican en el procedimiento siguiente para ver el conjunto de datos del proyecto usando la consola.

Para ver los conjuntos de datos (consola)

1. Abre la consola https://console.aws.amazon.com/lookoutvision/Amazon Lookout for Vision en.
2. Elija Comenzar.
3. En el panel de navegación izquierdo, elija Proyectos.
4. En la página Proyectos, elija el proyecto que contiene el conjunto de datos que desee visualizar.
5. En el panel de navegación izquierdo, elija Conjunto de datos para ver los detalles del conjunto de datos. Si tiene un conjunto de datos de entrenamiento y uno de prueba, se muestra una pestaña para cada conjunto de datos.

Visualización de los conjuntos de datos de un proyecto (SDK)

Puede utilizar la `DescribeDataset` operación para obtener información sobre el conjunto de datos de entrenamiento o de prueba asociado a un proyecto.

Para ver los conjuntos 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: Configura el AWS CLI y AWS SDKs.
2. Use el siguiente código de ejemplo para ver un conjunto de datos.

   CLI

   Cambie los siguientes valores:

   - `project-name` al nombre del proyecto que incluye el modelo que desea visualizar.

- `dataset-type` el tipo de conjunto de datos que desea visualizar (`train` o `test`).

```
aws lookoutvision describe-dataset --project-name project name\
  --dataset-type train or test \
  --profile lookoutvision-access
```

Python

Este código se ha tomado del GitHub repositorio de ejemplos del SDK de AWS documentación. Consulte el ejemplo completo aquí.

```
    @staticmethod
    def describe_dataset(lookoutvision_client, project_name, dataset_type):
        """
        Gets information about a Lookout for Vision dataset.

        :param lookoutvision_client: A Boto3 Lookout for Vision client.
        :param project_name: The name of the project that contains the dataset
 that
                             you want to describe.
        :param dataset_type: The type (train or test) of the dataset that you
 want
                             to describe.
        """
        try:
            response = lookoutvision_client.describe_dataset(
                ProjectName=project_name, DatasetType=dataset_type
            )
            print(f"Name: {response['DatasetDescription']['ProjectName']}")
            print(f"Type: {response['DatasetDescription']['DatasetType']}")
            print(f"Status: {response['DatasetDescription']['Status']}")
            print(f"Message: {response['DatasetDescription']['StatusMessage']}")
            print(f"Images: {response['DatasetDescription']['ImageStats']
['Total']}")
            print(f"Labeled: {response['DatasetDescription']['ImageStats']
['Labeled']}")
            print(f"Normal: {response['DatasetDescription']['ImageStats']
['Normal']}")
            print(f"Anomaly: {response['DatasetDescription']['ImageStats']
['Anomaly']}")
        except ClientError:
```

```
            logger.exception("Service error: problem listing datasets.")
            raise
        print("Done.")
```

Java V2

Este código se ha tomado del GitHub repositorio de ejemplos del SDK de AWS documentación. Consulte el ejemplo completo aquí.

```
/**
 * Gets the description for a Amazon Lookout for Vision dataset.
 *
 * @param lfvClient   An Amazon Lookout for Vision client.
 * @param projectName The name of the project in which you want to describe a
 *                    dataset.
 * @param datasetType The type of the dataset that you want to describe (train
 *                    or test).
 * @return DatasetDescription A description of the dataset.
 */
public static DatasetDescription describeDataset(LookoutVisionClient lfvClient,
                String projectName,
                String datasetType) throws LookoutVisionException {

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

        DescribeDatasetRequest describeDatasetRequest =
 DescribeDatasetRequest.builder()
                        .projectName(projectName)
                        .datasetType(datasetType)
                        .build();

        DescribeDatasetResponse describeDatasetResponse =
 lfvClient.describeDataset(describeDatasetRequest);
        DatasetDescription datasetDescription =
 describeDatasetResponse.datasetDescription();

        logger.log(Level.INFO, "Project: {0}\n"
                        + "Created: {1}\n"
                        + "Type: {2}\n"
                        + "Total: {3}\n"
                        + "Labeled: {4}\n"
```

```
                        + "Normal: {5}\n"
                        + "Anomalous: {6}\n",
                        new Object[] {
                                        datasetDescription.projectName(),
                                        datasetDescription.creationTimestamp(),
                                        datasetDescription.datasetType(),

 datasetDescription.imageStats().total().toString(),

 datasetDescription.imageStats().labeled().toString(),

 datasetDescription.imageStats().normal().toString(),

 datasetDescription.imageStats().anomaly().toString(),
                        });

        return datasetDescription;

}
```

Añadir imágenes a su conjunto de datos.

Después de crear un conjunto de datos, es posible que desee agregar más imágenes al conjunto de datos. Por ejemplo, si la evaluación del modelo indica que el modelo es deficiente, puede mejorar la calidad del modelo añadiendo más imágenes. Si ha creado un conjunto de datos de prueba, añadir más imágenes puede aumentar la precisión de las métricas de rendimiento del modelo.

Vuelva a entrenar el modelo después de actualizar los conjuntos de datos.

Temas

- Añadir más imágenes
- Cómo añadir más imágenes (SDK)

Añadir más imágenes

Puede añadir más imágenes a sus conjuntos de datos cargando imágenes desde su equipo local. Para añadir más imágenes etiquetadas con el SDK, utilice la UpdateDatasetEntriesoperación.

Cómo añadir más imágenes a su conjunto de datos (consola)

1. Elija Acciones y seleccione el conjunto de datos al que desee añadir las imágenes.
2. 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.
3. Seleccione Cargar imágenes.
4. Elija Guardar cambios.

Cuando termine de añadir más imágenes, tendrá que etiquetarlas para que puedan usarse para entrenar el modelo. Para obtener más información, consulte Clasificación de imágenes (consola).

Cómo añadir más imágenes (SDK)

Para añadir más imágenes etiquetadas con el SDK, utilice la UpdateDatasetEntriesoperación. Proporcione un archivo de manifiesto que contiene las imágenes que desea agregar. También puede actualizar las imágenes existentes especificando la imagen en el `source-ref` campo de la línea JSON del archivo de manifiesto. Para obtener más información, consulte Creación de un archivo de manifiesto.

Para añadir 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: Configura el AWS CLI y AWS SDKs.
2. Use el siguiente código de ejemplo para añadir más imágenes a un conjunto de datos.

   CLI

   Cambie los siguientes valores:

   - `project-name` al nombre del proyecto que incluye el conjunto de datos que desea actualizar.
   - `dataset-type` el tipo de conjunto de datos que desea actualizar (`train` o `test`).
   - `changes` a la ubicación del archivo de manifiesto que contiene las actualizaciones del conjunto de datos.

```
aws lookoutvision update-dataset-entries\
    --project-name project\
```

```
    --dataset-type train or test\
    --changes fileb://manifest file \
    --profile lookoutvision-access
```

Python

Este código se ha tomado del GitHub repositorio de ejemplos del SDK de AWS documentación. Consulte el ejemplo completo aquí.

```
    @staticmethod
    def update_dataset_entries(lookoutvision_client, project_name, dataset_type,
 updates_file):
        """
        Adds dataset entries to an Amazon Lookout for Vision dataset.
        :param lookoutvision_client: The Amazon Rekognition Custom Labels Boto3
 client.
        :param project_name: The project that contains the dataset that you want
 to update.
        :param dataset_type: The type of the dataset that you want to update
 (train or test).
        :param updates_file: The manifest file of JSON Lines that contains the
 updates.
        """

        try:
            status = ""
            status_message = ""
            manifest_file = ""

            # Update dataset entries
            logger.info(f"""Updating {dataset_type} dataset for project
 {project_name}
with entries from {updates_file}.""")

            with open(updates_file) as f:
                manifest_file = f.read()

            lookoutvision_client.update_dataset_entries(
                ProjectName=project_name,
                DatasetType=dataset_type,
                Changes=manifest_file,
            )
```

```
            finished = False
            while finished == False:

                dataset =
lookoutvision_client.describe_dataset(ProjectName=project_name,

DatasetType=dataset_type)

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

                if status == "UPDATE_IN_PROGRESS":
                    logger.info(
                        (f"Updating {dataset_type} dataset for project
{project_name}."))
                    time.sleep(5)
                    continue

                if status == "UPDATE_FAILED_ROLLBACK_IN_PROGRESS":
                    logger.info(
                        (f"Update failed, rolling back {dataset_type} dataset
for project {project_name}."))
                    time.sleep(5)
                    continue

                if status == "UPDATE_COMPLETE":
                    logger.info(
                        f"Dataset updated: {status} : {status_message} :
{dataset_type} dataset for project {project_name}.")
                    finished = True
                    continue

                if status == "UPDATE_FAILED_ROLLBACK_COMPLETE":
                    logger.info(
                        f"Rollback complated after update failure: {status} :
{status_message} : {dataset_type} dataset for project {project_name}.")
                    finished = True
                    continue

                logger.exception(
                    f"Failed. Unexpected state for dataset update: {status} :
{status_message} : {dataset_type} dataset for project {project_name}.")
                raise Exception(
```

```
                    f"Failed. Unexpected state for dataset update: {status} :
 {status_message} :{dataset_type} dataset for project {project_name}.")

            logger.info(f"Added entries to dataset.")

            return status, status_message

        except ClientError as err:
            logger.exception(
                f"Couldn't update dataset: {err.response['Error']['Message']}")
            raise
```

Java V2

Este código se ha tomado del GitHub repositorio de ejemplos del SDK de AWS documentación. Consulte el ejemplo completo aquí.

```
/**
 * Updates an Amazon Lookout for Vision dataset from a manifest file.
 * Returns after Lookout for Vision updates the dataset.
 *
 * @param lfvClient    An Amazon Lookout for Vision client.
 * @param projectName  The name of the project in which you want to update a
 *                     dataset.
 * @param datasetType  The type of the dataset that you want to update (train or
 *                     test).
 * @param manifestFile The name and location of a local manifest file that you
 want to
 * use to update the dataset.
 * @return DatasetStatus The status of the updated dataset.
 */

public static DatasetStatus updateDatasetEntries(LookoutVisionClient lfvClient,
 String projectName,
                String datasetType, String updateFile) throws
 FileNotFoundException, LookoutVisionException,
                InterruptedException {

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

        InputStream sourceStream = new FileInputStream(updateFile);
        SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);
```

```
       UpdateDatasetEntriesRequest updateDatasetEntriesRequest =
UpdateDatasetEntriesRequest.builder()
                       .projectName(projectName)
                       .datasetType(datasetType)
                       .changes(sourceBytes)
                       .build();

       lfvClient.updateDatasetEntries(updateDatasetEntriesRequest);

       boolean finished = false;
       DatasetStatus status = null;

       // Wait until update completes.

       do {

               DescribeDatasetRequest describeDatasetRequest =
DescribeDatasetRequest.builder()
                               .projectName(projectName)
                               .datasetType(datasetType)
                               .build();
               DescribeDatasetResponse describeDatasetResponse = lfvClient
                               .describeDataset(describeDatasetRequest);

               DatasetDescription datasetDescription =
describeDatasetResponse.datasetDescription();

               status = datasetDescription.status();

               switch (status) {

                       case UPDATE_COMPLETE:
                               logger.log(Level.INFO, "{0} Dataset updated for
project {1}.",
                                               new Object[] { datasetType,
projectName });
                               finished = true;
                               break;

                       case UPDATE_IN_PROGRESS:

                               logger.log(Level.INFO, "{0} Dataset update for
project {1} in progress.",
```

```
                                                new Object[] { datasetType,
 projectName });
                                TimeUnit.SECONDS.sleep(5);

                                break;

                        case UPDATE_FAILED_ROLLBACK_IN_PROGRESS:
                                logger.log(Level.SEVERE,
                                                "{0} Dataset update failed for
 project {1}. Rolling back",
                                                new Object[] { datasetType,
 projectName });

                                TimeUnit.SECONDS.sleep(5);

                                break;

                        case UPDATE_FAILED_ROLLBACK_COMPLETE:
                                logger.log(Level.SEVERE,
                                                "{0} Dataset update failed for
 project {1}. Rollback completed.",
                                                new Object[] { datasetType,
 projectName });
                                finished = true;
                                break;

                        default:
                                logger.log(Level.SEVERE,
                                                "{0} Dataset update failed for
 project {1}. Unexpected error returned.",
                                                new Object[] { datasetType,
 projectName });
                                finished = true;

                }

        } while (!finished);

        return status;

}
```

3. Repita el paso anterior y proporcione valores para el otro tipo de conjunto de datos.

Eliminar imágenes del conjunto de datos

No puede eliminar imágenes directamente de un conjunto de datos. En su lugar, debe eliminar el conjunto de datos existente y crear uno nuevo sin las imágenes que desea eliminar. La forma de eliminar las imágenes depende de cómo las haya importado al conjunto de datos existente (archivo de manifiesto, bucket de Amazon S3 o equipo local).

También puedes usar el AWS SDK para eliminar imágenes. Esto resulta útil cuando se crea un modelo de segmentación de imágenes sin un archivo de manifiesto de segmentación de imágenes, ya que no es necesario volver a dibujar las máscaras de las imágenes con la consola de Amazon Lookout for Vision.

Temas

- Eliminar imágenes de un conjunto de datos (consola)
- Eliminación de imágenes de un conjunto de datos (SDK)

Eliminar imágenes de un conjunto de datos (consola)

Utilice el siguiente procedimiento para eliminar imágenes de un conjunto de datos con la consola de Amazon Lookout for Vision.

Eliminar imágenes de un conjunto de datos (consola)

1. Abra la galería de conjuntos de datos del proyecto.
2. Anote el nombre de cada imagen que desea eliminar.
3. Elimine el conjunto de datos existente.
4. Realice una de las siguientes acciones:
   - Si creó el conjunto de datos con un archivo de manifiesto, haga lo siguiente:
     a. En un editor de texto, abra el archivo de manifiesto que usó para crear el conjunto de datos.
     b. Elimine la línea JSON de cada imagen que anotó en el paso 2. Puede identificar la línea JSON de una imagen marcando el campo `source-ref`.
     c. Guarde el archivo de manifiesto.

d. Cree un nuevo conjunto de datos con el archivo de manifiesto actualizado.

- Si creó el conjunto de datos a partir de imágenes importadas de un bucket de Amazon S3, haga lo siguiente:

  a. Elimine las imágenes que anotó en el paso 2 del bucket de Amazon S3.

  b. Cree un nuevo conjunto de datos con las imágenes restantes en el bucket de Amazon S3. Si clasifica las imágenes por nombre de carpeta, no necesitará clasificarlas en el siguiente paso.

  c. Realice una de las siguientes acciones:

     - Si va a crear un modelo de clasificación de imágenes, clasifique cada imagen sin etiquetar.
     - Si va a crear un modelo de segmentación de imágenes, clasifique y segmente cada imagen sin etiquetar.

- Si creó el conjunto de datos a partir de imágenes importadas de un equipo local, haga lo siguiente:

  a. En su equipo, cree una carpeta con las imágenes que quiera usar. No incluya las imágenes que desea eliminar del conjunto de datos. Para obtener más información, consulte Creación de un conjunto de datos con imágenes almacenadas en su equipo local.

  b. Cree el conjunto de datos con las imágenes en la carpeta que creó en el paso 4.a.

  c. Realice una de las siguientes acciones:

     - Si va a crear un modelo de clasificación de imágenes, clasifique cada imagen sin etiquetar.
     - Si va a crear un modelo de segmentación de imágenes, clasifique y segmente cada imagen sin etiquetar.

5. Entrenamiento del modelo.

Eliminación de imágenes de un conjunto de datos (SDK)

Puedes usar el AWS SDK para eliminar imágenes de un conjunto de datos.

Para eliminar imágenes de un conjunto de datos (SDK)

1. Abra la galería de conjuntos de datos del proyecto.

2. Anote el nombre de cada imagen que desea eliminar.
3. Exporta las líneas JSON del conjunto de datos mediante la ListDatasetEntriesoperación.
4. Cree un archivo de manifiesto con las líneas JSON exportadas.
5. En un editor de texto, abra el archivo de manifiesto.
6. Elimine la línea JSON de cada imagen que anotó en el paso 2. Puede identificar la línea JSON de una imagen marcando el campo `source-ref`.
7. Guarde el archivo de manifiesto.
8. Elimine el conjunto de datos existente.
9. Cree un nuevo conjunto de datos con el archivo de manifiesto actualizado.
10. Entrenamiento del modelo.

Eliminación de un conjunto de datos

Puede eliminar un conjunto de datos desde un proyecto usando la consola o mediante la operación `DeleteDataset`. Las imágenes a las que hace referencia un conjunto de datos no se eliminan. Si elimina el conjunto de datos de prueba de un proyecto que tiene un conjunto de datos de entrenamiento y otro de prueba, el proyecto vuelve a ser un proyecto de conjunto de datos único; el conjunto de datos restante se divide durante el entrenamiento para crear un conjunto de datos de entrenamiento y prueba. Si elimina el conjunto de datos de entrenamiento, no puede entrenar un proyecto hasta que cree un nuevo conjunto de datos de entrenamiento.

Eliminación de un conjunto de datos (consola)

Para eliminar un conjunto de datos, siga el procedimiento que se indica a continuación. Si elimina todos los conjuntos de datos de un proyecto, aparecerá la página Crear conjunto de datos.

Cómo eliminar un conjunto de datos (consola)

1. Abre la consola https://console.aws.amazon.com/lookoutvision/Amazon Lookout for Vision en.
2. Elija Comenzar.
3. En el panel de navegación izquierdo, elija Proyectos.
4. En la página Proyectos, elija el proyecto que contiene el conjunto de datos que desee eliminar.
5. En el panel de navegación izquierdo, elija Conjunto de datos.
6. Seleccione Acciones y a continuación el conjunto de datos que quiere eliminar.

7. En el cuadro de diálogo Eliminar, escriba eliminar para confirmar que desea eliminar el conjunto de datos.
8. Elija Eliminar conjunto de datos de entrenamiento o Eliminar conjunto de datos de prueba para eliminarlo.

Eliminación de un conjunto de datos (SDK)

Utilice la operación `DeleteDataset` para eliminar un conjunto de datos.

Para 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: Configura el AWS CLI y AWS SDKs.
2. Utilice el siguiente ejemplo para eliminar un modelo.

   CLI

   Cambie el valor de lo siguiente:

   - `project-name` al nombre del proyecto que incluye el modelo que desea eliminar.
   - `dataset-type` a uno `train` o a `test` otro, según el conjunto de datos que desee eliminar. Si tiene un proyecto de conjunto de datos único, especifique `train` eliminar el conjunto de datos.

   ```
   aws lookoutvision delete-dataset --project-name project name\
     --dataset-type dataset type \
     --profile lookoutvision-access
   ```

   Python

   Este código se ha tomado del GitHub repositorio de ejemplos del SDK de AWS documentación. Consulte el ejemplo completo aquí.

   ```
       @staticmethod
       def delete_dataset(lookoutvision_client, project_name, dataset_type):
           """
           Deletes a Lookout for Vision dataset
   ```

```
        :param lookoutvision_client: A Boto3 Lookout for Vision client.
        :param project_name: The name of the project that contains the dataset
 that
                             you want to delete.
        :param dataset_type: The type (train or test) of the dataset that you
                             want to delete.
        """
        try:
            logger.info(
                "Deleting the %s dataset for project %s.", dataset_type,
 project_name
            )
            lookoutvision_client.delete_dataset(
                ProjectName=project_name, DatasetType=dataset_type
            )
            logger.info("Dataset deleted.")
        except ClientError:
            logger.exception("Service error: Couldn't delete dataset.")
            raise
```

Java V2

Este código se ha tomado del GitHub repositorio de ejemplos del SDK de AWS documentación. Consulte el ejemplo completo aquí.

```
/**
 * Deletes the train or test dataset in an Amazon Lookout for Vision project.
 *
 * @param lfvClient   An Amazon Lookout for Vision client.
 * @param projectName The name of the project in which you want to delete a
 *                    dataset.
 * @param datasetType The type of the dataset that you want to delete (train or
 *                    test).
 * @return Nothing.
 */
public static void deleteDataset(LookoutVisionClient lfvClient, String
 projectName, String datasetType)
                throws LookoutVisionException {

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

```
        DeleteDatasetRequest deleteDatasetRequest =
 DeleteDatasetRequest.builder()
                        .projectName(projectName)
                        .datasetType(datasetType)
                        .build();

        lfvClient.deleteDataset(deleteDatasetRequest);

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

}
```

Exportación de conjuntos de datos de un proyecto (SDK)

Puede usar el AWS SDK para exportar conjuntos de datos de un proyecto de Amazon Lookout for Vision a una ubicación de bucket de Amazon S3.

Al exportar un conjunto de datos, puede realizar tareas como crear un proyecto de Lookout for Vision con una copia de los conjuntos de datos de un proyecto de origen. También puede crear una instantánea de los conjuntos de datos utilizados para una versión específica de un modelo.

El código Python de este procedimiento exporta el conjunto de datos de entrenamiento (imágenes del manifiesto y del conjunto de datos) de un proyecto a la ubicación de Amazon S3 de destino que especifique. Si está presente en el proyecto, el código también exporta el manifiesto del conjunto de datos de prueba y las imágenes del conjunto de datos. El destino puede estar en el mismo bucket de Amazon S3 que el proyecto de origen o en un bucket de Amazon S3 diferente. El código usa la ListDatasetEntriesoperación para obtener los archivos de manifiesto del conjunto de datos. Las operaciones de Amazon S3 copian las imágenes del conjunto de datos y los archivos de manifiesto actualizados en la ubicación de Amazon S3 de destino.

Este procedimiento muestra cómo exportar los conjuntos de datos de un proyecto. También se muestra cómo crear un nuevo proyecto con los conjuntos de datos exportados.

Para exportar conjuntos de datos de 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: Configura el AWS CLI y AWS SDKs.

2. Determine la ruta de Amazon S3 de destino para la exportación del conjunto de datos. Asegúrese de que el destino se encuentra en una AWS región compatible con Amazon Lookout for Vision. Si desea crear un bucket de Amazon S3, consulte Creación de un bucket.

3. Asegúrese de que el usuario tenga permisos de acceso a la ruta Amazon S3 de destino para la exportación del conjunto de datos y a las ubicaciones S3 de los archivos de imagen de los conjuntos de datos del proyecto de origen. Puede utilizar la siguiente política, que supone que los archivos de imágenes pueden estar en cualquier ubicación. *bucket/path*Sustitúyalo por el depósito y la ruta de destino para la exportación del conjunto de datos.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "PutExports",
            "Effect": "Allow",
            "Action": [
                "S3:PutObjectTagging",
                "S3:PutObject"
            ],
            "Resource": "arn:aws:s3:::bucket/path/*"
        },
        {
            "Sid": "GetSourceRefs",
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:GetObjectTagging",
                "s3:GetObjectVersion"
            ],
            "Resource": "*"
        }
    ]
}
```

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.

4. Guarde el siguiente código en un archivo denominado `dataset_export.py`.

```
"""
Purpose

Shows how to export the datasets (manifest files and images)
from an Amazon Lookout for Vision project to a new Amazon
S3 location.
"""

import argparse
import json
import logging

import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)


def copy_file(s3_resource, source_file, destination_file):
    """
    Copies a file from a source Amazon S3 folder to a destination
    Amazon S3 folder.
    The destination can be in a different S3 bucket.
    :param s3: An Amazon S3 Boto3 resource.
    :param source_file: The Amazon S3 path to the source file.
    :param destination_file: The destination Amazon S3 path for
    the copy operation.
    """
```

```
    source_bucket, source_key = source_file.replace("s3://", "").split("/", 1)
    destination_bucket, destination_key = destination_file.replace("s3://",
 "").split(
        "/", 1
    )

    try:
        bucket = s3_resource.Bucket(destination_bucket)
        dest_object = bucket.Object(destination_key)
        dest_object.copy_from(CopySource={"Bucket": source_bucket, "Key":
 source_key})
        dest_object.wait_until_exists()
        logger.info("Copied %s to %s", source_file, destination_file)
    except ClientError as error:
        if error.response["Error"]["Code"] == "404":
            error_message = (
                f"Failed to copy {source_file} to "
                f"{destination_file}. : {error.response['Error']['Message']}"
            )
            logger.warning(error_message)
            error.response["Error"]["Message"] = error_message
        raise


def upload_manifest_file(s3_resource, manifest_file, destination):
    """
    Uploads a manifest file to a destination Amazon S3 folder.
    :param s3: An Amazon S3 Boto3 resource.
    :param manifest_file: The manifest file that you want to upload.
    :destination: The Amazon S3 folder location to upload the manifest
    file to.
    """

    destination_bucket, destination_key = destination.replace("s3://",
 "").split("/", 1)

    bucket = s3_resource.Bucket(destination_bucket)

    put_data = open(manifest_file, "rb")
    obj = bucket.Object(destination_key + manifest_file)

    try:
        obj.put(Body=put_data)
        obj.wait_until_exists()
```

```
        logger.info("Put manifest file '%s' to bucket '%s'.", obj.key,
 obj.bucket_name)
    except ClientError:
        logger.exception(
            "Couldn't put manifest file '%s' to bucket '%s'.", obj.key,
 obj.bucket_name
        )
        raise
    finally:
        if getattr(put_data, "close", None):
            put_data.close()


def get_dataset_types(lookoutvision_client, project):
    """
    Determines the types of the datasets (train or test) in an
    Amazon Lookout for Vision project.
    :param lookoutvision_client: A Lookout for Vision Boto3 client.
    :param project: The Lookout for Vision project that you want to check.
    :return: The dataset types in the project.
    """

    try:
        response = lookoutvision_client.describe_project(ProjectName=project)

        datasets = []

        for dataset in response["ProjectDescription"]["Datasets"]:
            if dataset["Status"] in ("CREATE_COMPLETE", "UPDATE_COMPLETE"):
                datasets.append(dataset["DatasetType"])
        return datasets

    except lookoutvision_client.exceptions.ResourceNotFoundException:
        logger.exception("Project %s not found.", project)
        raise


def process_json_line(s3_resource, entry, dataset_type, destination):
    """
    Creates a JSON line for a new manifest file, copies image and mask to
    destination.
    :param s3_resource: An Amazon S3 Boto3 resource.
    :param entry: A JSON line from the manifest file.
    :param dataset_type: The type (train or test) of the dataset that
```

```
    you want to create the manifest file for.
    :param destination: The destination Amazon S3 folder for the manifest
    file and dataset images.
    :return: A JSON line with details for the destination location.
    """
    entry_json = json.loads(entry)

    print(f"source: {entry_json['source-ref']}")

    # Use existing folder paths to ensure console added image names don't clash.
    bucket, key = entry_json["source-ref"].replace("s3://", "").split("/", 1)
    logger.info("Source location: %s/%s", bucket, key)

    destination_image_location = destination + dataset_type + "/images/" + key

    copy_file(s3_resource, entry_json["source-ref"], destination_image_location)

    # Update JSON for writing.
    entry_json["source-ref"] = destination_image_location

    if "anomaly-mask-ref" in entry_json:
        source_anomaly_ref = entry_json["anomaly-mask-ref"]
        mask_bucket, mask_key = source_anomaly_ref.replace("s3://", "").split("/",
 1)

        destination_mask_location = destination + dataset_type + "/masks/" +
 mask_key
        entry_json["anomaly-mask-ref"] = destination_mask_location

        copy_file(s3_resource, source_anomaly_ref, entry_json["anomaly-mask-ref"])

    return entry_json


def write_manifest_file(
    lookoutvision_client, s3_resource, project, dataset_type, destination
):
    """
    Creates a manifest file for a dataset. Copies the manifest file and
    dataset images (and masks, if present) to the specified Amazon S3 destination.
    :param lookoutvision_client: A Lookout for Vision Boto3 client.
    :param project: The Lookout for Vision project that you want to use.
    :param dataset_type: The type (train or test) of the dataset that
    you want to create the manifest file for.
```

```
    :param destination: The destination Amazon S3 folder for the manifest file
    and dataset images.
    """

    try:
        # Create a reusable Paginator
        paginator = lookoutvision_client.get_paginator("list_dataset_entries")

        # Create a PageIterator from the Paginator
        page_iterator = paginator.paginate(
            ProjectName=project,
            DatasetType=dataset_type,
            PaginationConfig={"PageSize": 100},
        )

        output_manifest_file = dataset_type + ".manifest"

        # Create manifest file then upload to Amazon S3 with images.
        with open(output_manifest_file, "w", encoding="utf-8") as manifest_file:
            for page in page_iterator:
                for entry in page["DatasetEntries"]:
                    try:
                        entry_json = process_json_line(
                            s3_resource, entry, dataset_type, destination
                        )

                        manifest_file.write(json.dumps(entry_json) + "\n")

                    except ClientError as error:
                        if error.response["Error"]["Code"] == "404":
                            print(error.response["Error"]["Message"])
                            print(f"Excluded JSON line: {entry}")
                        else:
                            raise
        upload_manifest_file(
            s3_resource, output_manifest_file, destination + "datasets/"
        )

    except ClientError:
        logger.exception("Problem getting dataset_entries")
        raise


def export_datasets(lookoutvision_client, s3_resource, project, destination):
```

```
    """
    Exports the datasets from an Amazon Lookout for Vision project to a specified
    Amazon S3 destination.
    :param project: The Lookout for Vision project that you want to use.
    :param destination: The destination Amazon S3 folder for the exported datasets.
    """
    # Add trailing backslash, if missing.
    destination = destination if destination[-1] == "/" else destination + "/"

    print(f"Exporting project {project} datasets to {destination}.")

    # Get each dataset and export to destination.

    dataset_types = get_dataset_types(lookoutvision_client, project)
    for dataset in dataset_types:
        logger.info("Copying %s dataset to %s.", dataset, destination)

        write_manifest_file(
            lookoutvision_client, s3_resource, project, dataset, destination
        )

    print("Exported dataset locations")
    for dataset in dataset_types:
        print(f"   {dataset}: {destination}datasets/{dataset}.manifest")

    print("Done.")


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

    parser.add_argument("project", help="The project that contains the dataset.")
    parser.add_argument("destination", help="The destination Amazon S3 folder.")


def main():
    """
    Exports the datasets from an Amazon Lookout for Vision project to a
    destination Amazon S3 location.
    """
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")
```

```
    parser = argparse.ArgumentParser(usage=argparse.SUPPRESS)
    add_arguments(parser)

    args = parser.parse_args()

    try:
        session = boto3.Session(profile_name="lookoutvision-access")
        lookoutvision_client = session.client("lookoutvision")
        s3_resource = session.resource("s3")

        export_datasets(
            lookoutvision_client, s3_resource, args.project, args.destination
        )
    except ClientError as err:
        logger.exception(err)
        print(f"Failed: {format(err)}")


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

5. Ejecute el código. Indique los siguientes argumentos de línea de comandos:

   - project: el nombre del proyecto de origen que contiene los conjuntos de datos que desea exportar.
   - destino: la ruta Amazon S3 de destino para los conjuntos de datos.

   Por ejemplo, `python dataset_export.py myproject s3://bucket/path/` .

6. Anote las ubicaciones de los archivos de manifiesto en las que se muestra el código. Los necesitará en el paso 8.
7. Cree un nuevo proyecto de Lookout for Vision con el conjunto de datos exportado siguiendo las instrucciones que se encuentran en Creación de su proyecto.
8. Realice una de las siguientes acciones:

   - Use la consola de Lookout for Vision para crear conjuntos de datos para su nuevo proyecto siguiendo las instrucciones que se encuentran en Creación de un conjunto de datos con un archivo de manifiesto (consola). No es necesario realizar los pasos del 1 al 6.

     Para el paso 12, haga lo siguiente:

a. Si el proyecto de origen tiene un conjunto de datos de prueba, elija Separar conjuntos de datos de entrenamiento y de prueba; de lo contrario, elija un conjunto de datos individual.

b. Para la ubicación del archivo .manifest, introduzca la ubicación del archivo de manifiesto correspondiente (entrenamiento o prueba) que anotó en el paso 6.

- Utilice la CreateDatasetoperación para crear conjuntos de datos para su nuevo proyecto mediante el código que aparece enCreación de un conjunto de datos con un archivo de manifiesto (SDK). Para el parámetro `manifest_file`, use la ubicación del archivo de manifiesto que anotó en el paso 6. Si el proyecto de origen tiene un conjunto de datos de prueba, vuelva a usar el código para crear el conjunto de datos de prueba.

9. Si está listo, entrene el modelo siguiendo las instrucciones que aparecen en Entrenamiento del modelo.

Visualización de los modelos

Puede haber varias versiones de un modelo en un proyecto. También puede utilizar la consola para ver los modelos del proyecto. También puede utilizar la operación `ListModels`.

> **Note**
>
> Al final, la lista de proyectos es coherente. Si crea un modelo, puede que tenga que esperar un poco antes de que la lista de proyectos esté actualizada.

Visualización de los modelos (consola)

Realice los pasos que se indican en el procedimiento siguiente para ver el modelo del proyecto en la consola.

Para ver los modelos (consola)

1. Abre la consola https://console.aws.amazon.com/lookoutvision/Amazon Lookout for Vision en.
2. Elija Comenzar.
3. En el panel de navegación izquierdo, elija Proyectos.
4. En la página Proyectos, elija el proyecto que contiene los modelos que desee eliminar.

5. En el panel de navegación izquierdo, elija Modelos y a continuación, consulte los detalles del modelo.

Visualización de sus modelos (SDK)

Para ver las versiones de un modelo, utilice la operación `ListModels`. Para obtener información acerca de una versión de modelo específica, utilice la operación `DescribeModel`. En el siguiente ejemplo, se enumeran todas las versiones del modelo de un proyecto y, a continuación, se muestra la información sobre el rendimiento y la configuración de salida de las versiones individuales del modelo.

Para ver los modelos (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: Configura el AWS CLI y AWS SDKs.
2. Use el siguiente código de ejemplo para enumerar sus modelos y obtener información sobre un modelo.

   CLI

   Use el comando `list-models` para ver los proyectos en un proyecto.

   Cambie el siguientes valor:

   - `project-name` al nombre del proyecto que incluye el modelo que desea visualizar.

   ```
   aws lookoutvision list-models --project-name project name \
     --profile lookoutvision-access
   ```

   Para obtener más información sobre un modelo, use el comando `describe-model`. Cambie los siguientes valores:

   - `project-name` al nombre del proyecto que incluye el modelo que desea visualizar.
   - `model-version` a la versión del modelo que quiere describir.

   ```
   aws lookoutvision describe-model --project-name project name\
       --model-version model version \
   ```

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

Python

Este código se ha tomado del GitHub repositorio de ejemplos del SDK de AWS documentación. Consulte el ejemplo completo aquí.

```
    @staticmethod
    def describe_models(lookoutvision_client, project_name):
        """
        Gets information about all models in a Lookout for Vision project.

        :param lookoutvision_client: A Boto3 Lookout for Vision client.
        :param project_name: The name of the project that you want to use.
        """
        try:
            response =
lookoutvision_client.list_models(ProjectName=project_name)
            print("Project: " + project_name)
            for model in response["Models"]:
                Models.describe_model(
                    lookoutvision_client, project_name, model["ModelVersion"]
                )
                print()
            print("Done...")
        except ClientError:
            logger.exception("Couldn't list models.")
            raise
```

Java V2

Este código se ha tomado del GitHub repositorio de ejemplos del SDK de AWS documentación. Consulte el ejemplo completo aquí.

```
/**
 * Lists the models in an Amazon Lookout for Vision project.
 *
 * @param lfvClient   An Amazon Lookout for Vision client.
 * @param projectName The name of the project that contains the models that
 *                    you want to list.
 * @return List <Metadata> A list of models in the project.
```

```
 */
public static List<ModelMetadata> listModels(LookoutVisionClient lfvClient,
 String projectName)
                throws LookoutVisionException {

        ListModelsRequest listModelsRequest = ListModelsRequest.builder()
                        .projectName(projectName)
                        .build();

        // Get a list of models in the supplied project.
        ListModelsResponse response = lfvClient.listModels(listModelsRequest);

        for (ModelMetadata model : response.models()) {
                logger.log(Level.INFO, "Model ARN: {0}\nVersion: {1}\nStatus:
 {2}\nMessage: {3}", new Object[] {
                                model.modelArn(),
                                model.modelVersion(),
                                model.statusMessage(),
                                model.statusAsString() });
        }

        return response.models();

}
```

Eliminación de un modelo

Puede eliminar una versión de un modelo mediante la consola o la operación `DeleteModel`. No puede eliminar la versión del modelo que esté en ejecución o en fase de entrenamiento.

Si la versión del modelo está en ejecución, utilice primero la `StopModel` operación para detener la versión del modelo. Para obtener más información, consulte Paro del modelo Amazon Lookout for Vision. Si el modelo se está entrenando, espere a que termine antes de eliminarlo.

Puede que se tarde unos segundos en eliminar un modelo. Para determinar si se ha eliminado un modelo, llame ListProjectsy compruebe si la versión del modelo (`ModelVersion`) está en la `Models` matriz.

Eliminación de un modelo (consola)

Realice los siguientes pasos para eliminar un modelo de la consola.

Cómo eliminar un modelo (consola)

1. Abre la consola https://console.aws.amazon.com/lookoutvision/Amazon Lookout for Vision en.
2. Elija Comenzar.
3. En el panel de navegación izquierdo, elija Proyectos.
4. En la página Proyectos, elija el proyecto que contiene el modelo que desee eliminar.
5. En el panel de navegación izquierdo, elija Models (Modelos).
6. En la vista de modelos, seleccione el botón de opción correspondiente al modelo que desea eliminar.
7. En la parte superior de la página, elija Eliminar.
8. En el cuadro de diálogo Eliminar, escriba eliminar para confirmar que desea eliminar el modelo.
9. Seleccione Eliminar modelo para eliminar el modelo.

Eliminación de un modelo (SDK)

Utilice el siguiente procedimiento para eliminar el modelo con la operación `DeleteModel`.

Para 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: Configura el AWS CLI y AWS SDKs.
2. Utilice el siguiente ejemplo para eliminar un modelo.

   CLI

   Cambie los siguientes valores:

   - `project-name` al nombre del proyecto que incluye el modelo que desea eliminar.
   - `model-version` a la versión del modelo que desee eliminar.

```
aws lookoutvision delete-model --project-name project name\
  --model-version model version \
  --profile lookoutvision-access
```

Python

Este código se ha tomado del GitHub repositorio de ejemplos del SDK de AWS documentación. Consulte el ejemplo completo aquí.

```
    @staticmethod
    def delete_model(lookoutvision_client, project_name, model_version):
        """
        Deletes a Lookout for Vision model. The model must first be stopped and
can't
        be in training.

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

            model_exists = True
            while model_exists:
                response =
lookoutvision_client.list_models(ProjectName=project_name)

                model_exists = False
                for model in response["Models"]:
                    if model["ModelVersion"] == model_version:
                        model_exists = True

                if model_exists is False:
                    logger.info("Model deleted")
                else:
                    logger.info("Model is being deleted...")
                    time.sleep(2)

            logger.info("Deleted Model: %s", model_version)
        except ClientError:
            logger.exception("Couldn't delete model.")
            raise
```

Java V2

Este código se ha tomado del GitHub repositorio de ejemplos del SDK de AWS documentación. Consulte el ejemplo completo aquí.

```
/**
* Deletes an Amazon Lookout for Vision model.
*
* @param lfvClient    An Amazon Lookout for Vision client. Returns after the
 model is deleted.
* @param projectName  The name of the project that contains the model that you
 want to delete.
* @param modelVersion The version of the model that you want to delete.
* @return void
*/
public static void deleteModel(LookoutVisionClient lfvClient,
                String projectName,
                String modelVersion) throws LookoutVisionException,
 InterruptedException {

        DeleteModelRequest deleteModelRequest = DeleteModelRequest.builder()
                        .projectName(projectName)
                        .modelVersion(modelVersion)
                        .build();

        lfvClient.deleteModel(deleteModelRequest);

        boolean deleted = false;

        do {

                ListModelsRequest listModelsRequest =
 ListModelsRequest.builder()
                                .projectName(projectName)
                                .build();

                // Get a list of models in the supplied project.
                ListModelsResponse response =
 lfvClient.listModels(listModelsRequest);
```

```
                ModelMetadata modelMetadata = response.models().stream()
                                .filter(model ->
 model.modelVersion().equals(modelVersion)).findFirst()
                                .orElse(null);

                if (modelMetadata == null) {
                        deleted = true;
                        logger.log(Level.INFO, "Deleted: Model version {0} of
 project {1}.",
                                        new Object[] { modelVersion,
 projectName });

                } else {
                        logger.log(Level.INFO, "Not yet deleted: Model version
 {0} of project {1}.",
                                        new Object[] { modelVersion,
 projectName });
                        TimeUnit.SECONDS.sleep(60);
                }

        } while (!deleted);

}
```

Etiquetado de modelos

Puede usar etiquetas para identificar, organizar, buscar y filtrar sus modelos de Etiquetas personalizadas de Amazon Lookout for Vision. 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, podría etiquetarlos con una clave de `Cost center` y añada 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 del acceso a los 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 Amazon Lookout for Vision formado.

Puede etiquetar modelos con la consola Amazon Lookout for Vision o con AWS SDKs la.

Temas

- Etiquetado de modelos (consola)
- Etiquetado de modelos (SDK)

Etiquetado de modelos (consola)

Puede usar la consola de Amazon Lookout for Vision para añadir etiquetas a los modelos, ver las etiquetas asociadas a un modelo y eliminar etiquetas.

Adición o eliminación de etiquetas (consola)

Este procedimiento explica cómo añadir o eliminar etiquetas en un modelo existente. También puede añadir etiquetas a un nuevo modelo cuando esté entrenado. Para obtener más información, consulte Entrenamiento del modelo.

Cómo añadir o eliminar etiquetas en un modelo existente (consola)

1. Abre la consola https://console.aws.amazon.com/lookoutvision/Amazon Lookout for Vision en.
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 (consola)

Puede usar la consola de Amazon Lookout for Vision 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 (SDK).

Para ver las etiquetas asociadas a un modelo

1. Abre la consola https://console.aws.amazon.com/lookoutvision/Amazon Lookout for Vision en.
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 AWS SDK para:

- Añadir etiquetas a un nuevo modelo
- Añadir etiquetas a un modelo existente
- Ver las etiquetas asociadas a un modelo
- Eliminar etiquetas de un modelo

En esta sección se incluyen AWS CLI ejemplos. Si no ha instalado el AWS CLI, consulte Paso 4: Configura el AWS CLI y AWS SDKs.

Cómo añadir etiquetas a un nuevo modelo (SDK)

Puede añadir etiquetas a un modelo al crearlo mediante la CreateModeloperación. Indique una o varias etiquetas en el parámetro de entrada `Tags` de la matriz.

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

Para obtener información sobre cómo crear y entrenar un modelo, consulte Entrenamiento de un modelo (SDK).

Cómo añadir etiquetas a un modelo existente (SDK)

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.

```
aws lookoutvision tag-resource --resource-arn "resource-arn"\
  --tags '[{"Key":"Key","Value":"Value"}]' \
  --profile lookoutvision-access
```

Para ver un ejemplo de código Java, consulte TagModel.

Listado de etiquetas de modelos (SDK)

Para enumerar las etiquetas adjuntas a un modelo, utilice la ListTagsForResourceoperación y especifique el nombre de recurso de Amazon (ARN) del modelo, the (`ResourceArn`). El resultado será la asignación de las claves y los valores de las etiquetas que se asocian al modelo concreto.

```
aws lookoutvision list-tags-for-resource --resource-arn resource-arn \
  --profile lookoutvision-access
```

Para ver qué modelos de un proyecto tienen una etiqueta específica, llame a `ListModels` para obtener una lista de modelos. A continuación, llame a `ListTagsForResource` para cada modelo en la respuesta a través de `ListModels`. Revise la respuesta de `ListTagsForResource` para ver si está la etiqueta necesaria.

Para ver un ejemplo de código Java, consulte ListModelTags. Para ver un ejemplo de código Python que busca un valor de etiqueta en todos los proyectos, consulte find_tag.py.

Eliminación de etiquetas de un modelo (SDK)

Para eliminar una o más etiquetas de un modelo, utilice la UntagResourceoperación. Indique el nombre de recurso de Amazon (ARN) del modelo (`ResourceArn`) y las etiquetas (`Tag-Keys`) que desea eliminar.

```
aws lookoutvision untag-resource --resource-arn resource-arn\
  --tag-keys '["Key"]' \
  --profile lookoutvision-access
```

Para ver un ejemplo de código Java, consulte UntagModel.

Visualización de las tareas de pruebas de detección

Puede ver las pruebas de detección mediante la consola. No puede usar el AWS SDK para ver las tareas de detección de pruebas.

> **Note**
>
> En última instancia, la lista de pruebas de detección es coherente. Si crea una prueba de detección, puede que tenga que esperar un poco antes de que la lista de pruebas de detección esté actualizada.

Visualización de las tareas de pruebas de detección (consola)

Utilice los siguientes procedimientos para ver las pruebas de detección.

Para ver las tareas de pruebas de detección

1. Abre la consola https://console.aws.amazon.com/lookoutvision/Amazon Lookout for Vision en.
2. Elija Comenzar.
3. En el panel de navegación izquierdo, elija Detecciones de prueba.
4. En la página de detecciones de ensayos, elija una tarea de pruebas de detección para ver sus detalles.

Ejemplos de código y conjuntos de datos

A continuación, se muestran ejemplos de código y conjuntos de datos que puede utilizar con Amazon Lookout for Vision.

Temas

Código de ejemplo

Están disponibles los siguientes ejemplos de código de Amazon Lookout for Vision.

Ejemplo	Descripción
GitHub	Ejemplo de código Python que forma y aloja un modelo de Amazon Lookout for Vision.
Amazon Lookout for Vision Lab	Un cuaderno de Python que puede usar para crear un modelo con las imágenes de ejemplo de la placa de circuito.
Ejemplo de código de python	Ejemplos de Python utilizados en la documenta ción de Amazon Lookout for Vision.
Código de ejemplo de Java	Ejemplos de Java utilizados en la documenta ción de Amazon Lookout for Vision.

Conjuntos de datos de ejemplo

Los siguientes son ejemplos de conjuntos de datos que puede usar con Amazon Lookout for Vision.

Temas

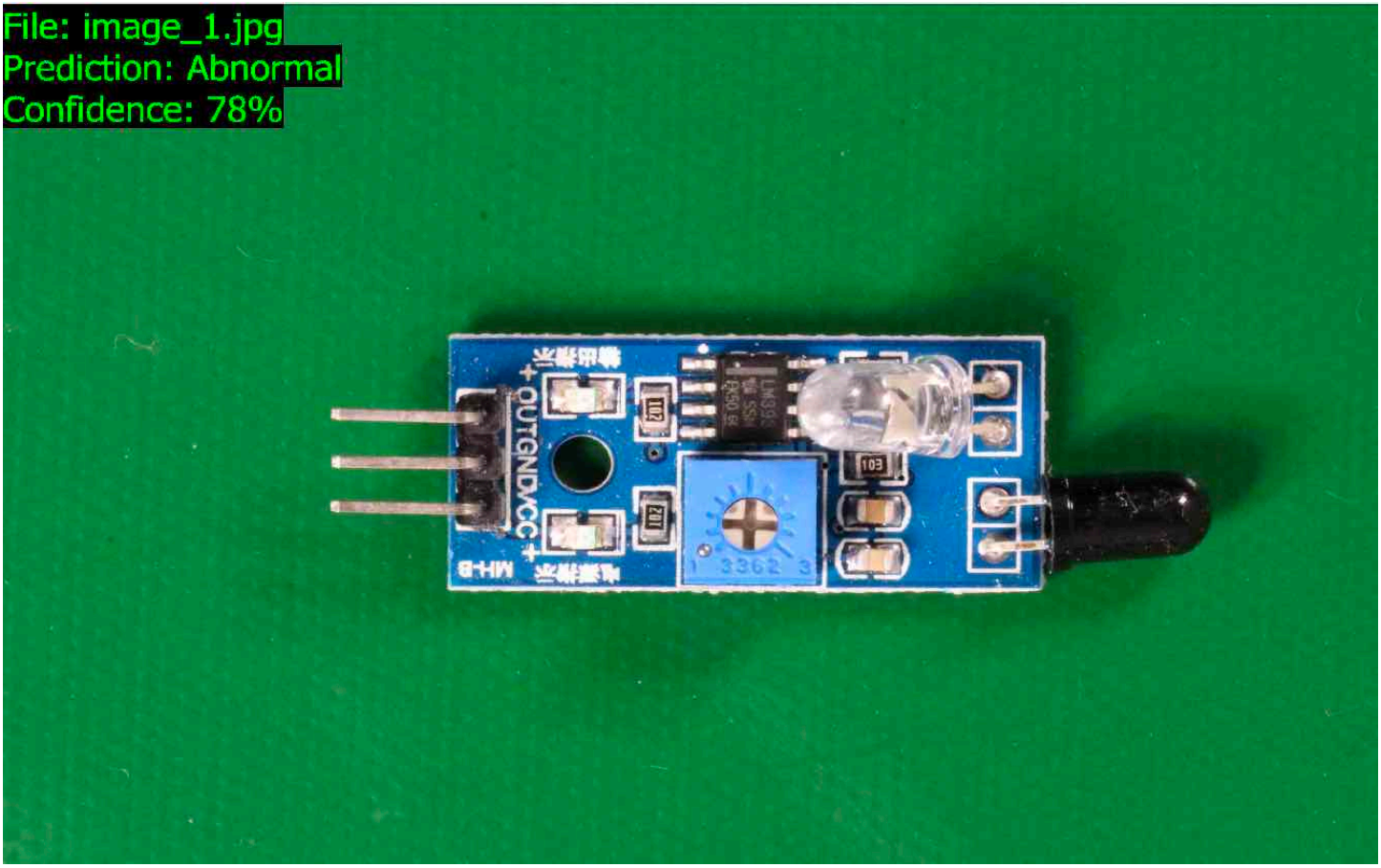
Conjuntos de datos de segmentación de imagen

Introducción a Amazon Lookout for Vision proporciona un conjunto de datos de cookies rotas que puede utilizar para crear un modelo de segmentación de imagen.

Para ver otro conjunto de datos que cree un modelo de segmentación de imágenes, consulte Identificar la ubicación de anomalías con Amazon Lookout for Vision en el borde sin usar una GPU.

Conjunto de datos de clasificación de imágenes

Amazon Lookout for Vision proporciona imágenes de ejemplo de placas de circuitos que puede utilizar para crear un modelo de clasificación de imágenes.



Puede copiar las imágenes del https://github.com/aws-samples/amazon-lookout-for-vision GitHubrepositorio. Las imágenes están en la carpeta `circuitboard`.

La carpeta `circuitboard` tiene las siguientes carpetas.

- `train`: imágenes que puede usar en un conjunto de datos de entrenamiento.

- `test`: imágenes que puede usar en un conjunto de datos.
- `extra_images`— Imágenes que puede utilizar para realizar un ensayo de detección o para probar su modelo entrenado durante la DetectAnomaliesoperación.

Las carpetas `train` y `test` tienen una subcarpeta denominada `normal` (contiene imágenes que son normales) y una subcarpeta denominada `anomaly` (contiene imágenes con anomalías).

> **Note**
>
> Más adelante, al crear un conjunto de datos con la consola, Amazon Lookout for Vision puede usar los nombres de las carpetas (`normal` y `anomaly`) para etiquetar las imágenes automáticamente. Para obtener más información, consulte the section called “Bucket de Amazon S3”.

Preparar las imágenes del conjunto de datos

1. Clona el https://github.com/aws-samples/amazon-lookout-for-visionrepositorio en tu ordenador. Para obtener más información, consulte Clonación de un repositorio.
2. Crear un bucket de Amazon S3. Para obtener más información, consulte ¿Cómo puedo crear un bucket de S3?
3. En el símbolo del sistema, introduzca el siguiente comando para copiar las imágenes del conjunto de datos de su equipo a su bucket de Amazon S3.

   ```
   aws s3 cp --recursive your-repository-folder/circuitboard s3://your-bucket/
   circuitboard
   ```

Tras cargar las imágenes, puede crear un modelo. Puede clasificar automáticamente las imágenes añadiendo las imágenes de la ubicación de Amazon S3 en la que cargó previamente las imágenes de la placa de circuito. Recuerde que se le cobrará por cada entrenamiento exitosa de un modelo y por el tiempo que el modelo esté funcionando (en host).

Crear un modelo de clasificación

1. Realice Creación de un proyecto (consola).

2. Realice Creación de un conjunto de datos con imágenes almacenadas en un bucket de Amazon S3.

   - Para el paso 6, elija la pestaña Separar los conjuntos de datos de entrenamiento y prueba.
   - En el paso 8a, introduzca el URI de S3 de las imágenes de entrenamiento que ha cargado para preparar las imágenes del conjunto de datos. Por ejemplo, `s3://your-bucket/circuitboard/train`. En el paso 8b, ingrese el URI de S3 correspondiente al conjunto de datos de prueba. Por ejemplo, `s3://your-bucket/circuitboard/test`.
   - Asegúrese de realizar el paso 9.
3. Realice Entrenamiento de un modelo (consola).
4. Realice Iniciar el modelo (consola).
5. Realice Detección de anomalías en una imagen. Puede utilizar imágenes de la carpeta `test_images`.
6. Cuando haya terminado con el modelo, realice Detención de un modelo (consola).

Seguridad en Amazon Lookout for Vision

La seguridad en la nube AWS es la máxima prioridad. Como AWS cliente, usted se beneficia de los centros de datos y las arquitecturas de red diseñados para cumplir con los requisitos de las organizaciones más sensibles a la seguridad.

La seguridad es una responsabilidad compartida entre AWS usted y usted. El modelo de responsabilidad compartida la describe como seguridad de la nube y seguridad en la nube:

- Seguridad de la nube: AWS es responsable de proteger la infraestructura que ejecuta AWS los servicios en la AWS nube. AWS también le proporciona servicios que puede utilizar de forma segura. Los auditores externos prueban y verifican periódicamente la eficacia de nuestra seguridad como parte de los AWS programas de de . Para obtener información sobre los programas de conformidad que se aplican a Amazon Lookout for Vision, consulte Servicios de AWS en el ámbito del programa de conformidad.
- Seguridad en la nube: su responsabilidad viene determinada por el AWS servicio que utilice. También eres responsable de otros factores, incluida la confidencialidad de los datos, los requisitos de la empresa y la legislación y la normativa aplicables.

Esta documentación le ayuda a comprender cómo aplicar el modelo de responsabilidad compartida cuando se utiliza Amazon Lookout for Vision. En los siguientes temas, se le mostrará cómo configurar Lookout for Vision para satisfacer sus objetivos de seguridad y conformidad. También aprenderá a utilizar otros servicios de AWS que le ayudarán a monitorear y a proteger los recursos de Lookout for Vision.

Temas

Protección de datos en Amazon Lookout for Vision

El modelo de se aplica a protección de datos en Amazon Lookout for Vision. Como se describe en este modelo, AWS es responsable de proteger la infraestructura global en la que se ejecutan todos los Nube de AWS. Eres responsable de mantener el control sobre el contenido alojado en esta infraestructura. También eres responsable de las tareas de administración y configuración de seguridad para los Servicios de AWS que utiliza. Para obtener más información sobre la privacidad de los datos, consulta las Preguntas frecuentes sobre la privacidad de datos. Para obtener información sobre la protección de datos en Europa, consulta la publicación de blog sobre el Modelo de responsabilidad compartida de AWS y GDPR en el Blog de seguridad de AWS .

Con fines de protección de datos, le recomendamos que proteja Cuenta de AWS las credenciales y configure los usuarios individuales con AWS IAM Identity Center o AWS Identity and Access Management (IAM). De esta manera, solo se otorgan a cada usuario los permisos necesarios para cumplir sus obligaciones laborales. También recomendamos proteger sus datos de la siguiente manera:

- Utiliza la autenticación multifactor (MFA) en cada cuenta.
- Utilice SSL/TLS para comunicarse con los recursos. AWS Se recomienda el uso de TLS 1.2 y recomendamos TLS 1.3.
- Configure la API y el registro de actividad de los usuarios con. AWS CloudTrail Para obtener información sobre el uso de CloudTrail senderos para capturar AWS actividades, consulte Cómo trabajar con CloudTrail senderos en la Guía del AWS CloudTrail usuario.
- Utilice soluciones de AWS cifrado, junto con todos los controles de seguridad predeterminados Servicios de AWS.
- Utiliza servicios de seguridad administrados avanzados, como Amazon Macie, que lo ayuden a detectar y proteger los datos confidenciales almacenados en Amazon S3.
- Si necesita módulos criptográficos validados por FIPS 140-3 para acceder a AWS través de una interfaz de línea de comandos o una API, utilice un punto final FIPS. Para obtener más información sobre los puntos de conexión de FIPS disponibles, consulta Estándar de procesamiento de la información federal (FIPS) 140-3.

Se recomienda encarecidamente no introducir nunca información confidencial o sensible, como por ejemplo, direcciones de correo electrónico de clientes, en etiquetas o campos de formato libre, tales como el campo Nombre. Esto incluye cuando trabajas con Lookout for Vision u Servicios de AWS otro mediante la consola, la API AWS CLI o. AWS SDKs Cualquier dato que ingrese en

etiquetas o campos de texto de formato libre utilizados para nombres se puede emplear para los registros de facturación o diagnóstico. Si proporciona una URL a un servidor externo, recomendamos encarecidamente que no incluya información de credenciales en la URL a fin de validar la solicitud para ese servidor.

Cifrado de datos

La siguiente información explica dónde Amazon Lookout for Vision utiliza el cifrado de datos para proteger los datos.

Cifrado en reposo

Imágenes

Para entrenar el modelo, Amazon Lookout for Vision hace una copia de las imágenes de origen de entrenamiento y prueba. Las imágenes copiadas se cifran en reposo en Amazon Simple Storage Service (S3) mediante el cifrado del lado del servidor con Clave propiedad de AWS una o más claves que usted proporciona. Las claves se almacenan mediante AWS Key Management Service (SSE-KMS). Las imágenes de origen no se ven afectadas. Para obtener más información, consulte Entrenamiento del modelo.

Modelos de Amazon Lookout for Vision

De forma predeterminada, los modelos entrenados y los archivos de manifiesto se cifran en Amazon S3 utilizando el cifrado del servidor con claves KMS en AWS Key Management Service (SSE-KMS). Lookout for Vision utiliza una Clave propiedad de AWS. Para obtener más información, consulte Protección de los datos con el cifrado del servidor. Los resultados del entrenamiento se escriben en el bucket especificado en el parámetro de entrada `output_bucket` para `CreateModel`. Los resultados del entrenamiento se cifran mediante los ajustes de cifrado configurados para el bucket (`output_bucket`).

Bucket de consola de Amazon Lookout for Vision

La consola de Amazon Lookout for Vision crea un bucket de Amazon S3 (bucket de consola) que puede utilizar para administrar sus proyectos. El bucket de la consola se cifra con el cifrado predeterminado de Amazon S3. Para obtener más información, consulte Servicio de almacenamiento simple cifrado predeterminado de Amazon para buckets de S3. Si utiliza su propia clave KMS, configure el bucket de la consola tras crearlo. Para obtener más información, consulte Protección de los datos con el cifrado del servidor. Amazon Lookout for Vision bloquea el acceso público al bucket de la consola.

Cifrado en tránsito

Los puntos de conexión de API de Amazon Lookout for Vision solo admiten conexiones seguras a través de HTTPS. Toda la comunicación está cifrada con Transport Layer Security (TLS).

Administración de claves

Puede utilizar AWS Key Management Service (KMS) para administrar las claves de las imágenes de entrada y los vídeos que almacena en los buckets de Amazon S3. Para obtener más información, consulte Paso 5: (opcional) Usar su propia clave de AWS Key Management Service.

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 clave de AWS Key Management Service (KMS). Para obtener más información, consulte Conceptos de AWS Key Management Service.

Privacidad del tráfico entre redes

El punto de conexión de Amazon Virtual Private Cloud (Amazon VPC) para Amazon Lookout for Vision es una entidad lógica dentro de una VPC que permite la conectividad solo a Amazon Lookout for Vision. Amazon VPC enruta las solicitudes a Amazon Lookout for Vision y vuelve a enrutar las respuestas a la VPC. Para obtener más información, consulte Puntos de conexión de VPC en la Guía del usuario de Amazon VPC. Para obtener información sobre el uso de puntos de conexión de Amazon VPC con Amazon Lookout for Vision, consulte Acceder a Amazon Lookout for Vision mediante un punto de conexión de interfaz (AWS PrivateLink).

Administración de identidades y accesos para Amazon Lookout for Vision

AWS Identity and Access Management (IAM) es una Servicio de AWS que ayuda al administrador a controlar de forma segura el acceso a AWS los recursos. Los administradores de IAM controlan quién puede estar autenticado (ha iniciado sesión) y autorizado (tiene permisos) para utilizar recursos de Lookout for Vision. La IAM es una Servicio de AWS herramienta que puede utilizar sin coste adicional.

Temas

- Cómo funciona Amazon Lookout for Vision con IAM
- Ejemplos de políticas basadas en identidades para Amazon Lookout for Vision
- AWS políticas gestionadas para Amazon Lookout for Vision
- Solución de problemas de identidad y acceso de Amazon Lookout for Vision

Público

La forma de usar AWS Identity and Access Management (IAM) varía según el trabajo que realices en Lookout for Vision.

Usuario de servicio: si utiliza el servicio de Lookout for Vision para realizar su trabajo, su administrador le proporciona las credenciales y los permisos que necesita. A medida que utilice más características de Lookout for Vision para realizar su trabajo, es posible que necesite permisos adicionales. Entender cómo se administra el acceso puede ayudarle a solicitar los permisos correctos al administrador. Si no puede acceder a una característica en Amazon Lookout for Vision, consulte Solución de problemas de identidad y acceso de Amazon Lookout for Vision.

Administrador de servicio: si está a cargo de los recursos de Lookout for Vision en su empresa, probablemente tenga acceso completo a Lookout for Vision. Su trabajo consiste en determinar a qué características y recursos de Lookout for Vision deben acceder los usuarios del servicio. Luego, debe enviar solicitudes a su gestionador de IAM para cambiar los permisos de los usuarios de su servicio. Revise la información de esta página para conocer los conceptos básicos de IAM. Para obtener más información sobre cómo su empresa puede utilizar IAM con Lookout for Vision, consulte Cómo funciona Amazon Lookout for Vision con IAM.

Administrador de IAM: si es un administrador de IAM, es posible que quiera conocer más detalles sobre cómo escribir políticas para administrar el acceso a Lookout for Vision. Para consultar ejemplos de políticas basadas en la identidad de Lookout for Vision que puede utilizar en IAM, consulte Ejemplos de políticas basadas en identidades para Amazon Lookout for Vision.

Autenticación con identidades

La autenticación es la forma en que inicias sesión AWS con tus credenciales de identidad. Debe estar autenticado (con quien haya iniciado sesión AWS) como usuario de IAM o asumiendo una función de IAM. Usuario raíz de la cuenta de AWS

Puede iniciar sesión AWS como una identidad federada mediante las credenciales proporcionadas a través de una fuente de identidad. AWS IAM Identity Center Los usuarios (IAM Identity Center), la

autenticación de inicio de sesión único de su empresa y sus credenciales de Google o Facebook son ejemplos de identidades federadas. Al iniciar sesión como una identidad federada, su gestionador habrá configurado previamente la federación de identidades mediante roles de IAM. Cuando accedes AWS mediante la federación, estás asumiendo un rol de forma indirecta.

Según el tipo de usuario que sea, puede iniciar sesión en el portal AWS Management Console o en el de AWS acceso. Para obtener más información sobre cómo iniciar sesión AWS, consulte Cómo iniciar sesión Cuenta de AWS en su Guía del AWS Sign-In usuario.

Si accede AWS mediante programación, AWS proporciona un kit de desarrollo de software (SDK) y una interfaz de línea de comandos (CLI) para firmar criptográficamente sus solicitudes con sus credenciales. Si no utilizas AWS herramientas, debes firmar las solicitudes tú mismo. Para obtener más información sobre la firma de solicitudes, consulte AWS Signature Versión 4 para solicitudes API en la Guía del usuario de IAM.

Independientemente del método de autenticación que use, es posible que deba proporcionar información de seguridad adicional. Por ejemplo, le AWS recomienda que utilice la autenticación multifactor (MFA) para aumentar la seguridad de su cuenta. Para obtener más información, consulte Autenticación multifactor en la Guía del usuario de AWS IAM Identity Center y Autenticación multifactor AWS en IAM en la Guía del usuario de IAM.

Cuenta de AWS usuario root

Al crear una Cuenta de AWS, comienza con una identidad de inicio de sesión que tiene acceso completo a todos Servicios de AWS los recursos de la cuenta. Esta identidad se denomina usuario Cuenta de AWS raíz y se accede a ella iniciando sesión con la dirección de correo electrónico y la contraseña que utilizaste para crear la cuenta. Recomendamos encarecidamente que no utiliza el usuario raíz para sus tareas diarias. Proteja las credenciales del usuario raíz y utilícelas solo para las tareas que solo el usuario raíz pueda realizar. Para ver la lista completa de las tareas que requieren que inicie sesión como usuario raíz, consulta Tareas que requieren credenciales de usuario raíz en la Guía del usuario de IAM.

Identidad federada

Como práctica recomendada, exija a los usuarios humanos, incluidos los que requieren acceso de administrador, que utilicen la federación con un proveedor de identidades para acceder Servicios de AWS mediante credenciales temporales.

Una identidad federada es un usuario del directorio de usuarios empresarial, un proveedor de identidades web AWS Directory Service, el directorio del Centro de Identidad o cualquier usuario al

que acceda Servicios de AWS mediante las credenciales proporcionadas a través de una fuente de identidad. Cuando las identidades federadas acceden Cuentas de AWS, asumen funciones y las funciones proporcionan credenciales temporales.

Para una administración de acceso centralizada, le recomendamos que utiliza AWS IAM Identity Center. Puede crear usuarios y grupos en el Centro de identidades de IAM o puede conectarse y sincronizarse con un conjunto de usuarios y grupos de su propia fuente de identidad para usarlos en todas sus Cuentas de AWS aplicaciones. Para obtener más información, consulta ¿Qué es el Centro de identidades de IAM? en la Guía del usuario de AWS IAM Identity Center .

Usuarios y grupos de IAM

Un usuario de IAM es una identidad propia Cuenta de AWS que tiene permisos específicos para una sola persona o aplicación. Siempre que sea posible, recomendamos emplear credenciales temporales, en lugar de crear usuarios de IAM que tengan credenciales de larga duración como contraseñas y claves de acceso. No obstante, si tiene casos de uso específicos que requieran credenciales de larga duración con usuarios de IAM, recomendamos rotar las claves de acceso. Para más información, consulta Rotar las claves de acceso periódicamente para casos de uso que requieran credenciales de larga duración en la Guía del usuario de IAM.

Un grupo de IAM es una identidad que especifica un conjunto de usuarios de IAM. No puedes iniciar sesión como grupo. Puedes usar los grupos para especificar permisos para varios usuarios a la vez. Los grupos facilitan la administración de los permisos para grandes conjuntos de usuarios. Por ejemplo, puede asignar un nombre a un grupo IAMAdminsy concederle permisos para administrar los recursos de IAM.

Los usuarios son diferentes de los roles. Un usuario se asocia exclusivamente a una persona o aplicación, pero la intención es que cualquier usuario pueda asumir un rol que necesite. Los usuarios tienen credenciales de larga duración permanentes; no obstante, los roles proporcionan credenciales temporales. Para obtener más información, consulte Casos de uso para usuarios de IAM en la Guía del usuario de IAM.

Roles de IAM

Un rol de IAM es una identidad dentro de usted Cuenta de AWS que tiene permisos específicos. Es similar a un usuario de IAM, pero no está asociado a una persona determinada. Para asumir temporalmente un rol de IAM en el AWS Management Console, puede cambiar de un rol de usuario a uno de IAM (consola). Puedes asumir un rol llamando a una operación de AWS API AWS CLI

o usando una URL personalizada. Para más información sobre los métodos para el uso de roles, consulta Métodos para asumir un rol en la Guía del usuario de IAM.

Los roles de IAM con credenciales temporales son útiles en las siguientes situaciones:

- Acceso de usuario federado: para asignar permisos a una identidad federada, puedes crear un rol y definir sus permisos. Cuando se autentica una identidad federada, se asocia la identidad al rol y se le conceden los permisos define el rol. Para obtener información acerca de roles de federación, consulte Crear un rol para un proveedor de identidad de terceros (federación) en la Guía de usuario de IAM. Si utiliza el IAM Identity Center, debe configurar un conjunto de permisos. IAM Identity Center correlaciona el conjunto de permisos con un rol en IAM para controlar a qué puedes acceder las identidades después de autenticarse. Para obtener información acerca de los conjuntos de permisos, consulta Conjuntos de permisos en la Guía del usuario de AWS IAM Identity Center .
- Permisos de usuario de IAM temporales: un usuario de IAM puedes asumir un rol de IAM para recibir temporalmente permisos distintos que le permitan realizar una tarea concreta.
- Acceso entre cuentas: puedes utilizar un rol de IAM para permitir que alguien (una entidad principal de confianza) de otra cuenta acceda a los recursos de la cuenta. Los roles son la forma principal de conceder acceso entre cuentas. Sin embargo, con algunas Servicios de AWS, puedes adjuntar una política directamente a un recurso (en lugar de usar un rol como proxy). Para obtener información acerca de la diferencia entre los roles y las políticas basadas en recursos para el acceso entre cuentas, consulta Acceso a recursos entre cuentas en IAM en la Guía del usuario de IAM.
- Acceso entre servicios: algunos Servicios de AWS utilizan funciones en otros Servicios de AWS. Por ejemplo, cuando realizas una llamada en un servicio, es habitual que ese servicio ejecute aplicaciones en Amazon EC2 o almacene objetos en Amazon S3. Es posible que un servicio haga esto usando los permisos de la entidad principal, usando un rol de servicio o usando un rol vinculado al servicio.
  - Sesiones de acceso directo (FAS): cuando utilizas un usuario o un rol de IAM para realizar acciones en AWS ellas, se te considera principal. Cuando utiliza algunos servicios, es posible que realice una acción que desencadene otra acción en un servicio diferente. El FAS utiliza los permisos del principal que llama Servicio de AWS y los solicita Servicio de AWS para realizar solicitudes a los servicios descendentes. Las solicitudes de FAS solo se realizan cuando un servicio recibe una solicitud que requiere interacciones con otros Servicios de AWS recursos para completarse. En este caso, debe tener permisos para realizar ambas acciones. Para

obtener información sobre las políticas a la hora de realizar solicitudes de FAS, consulta Reenviar sesiones de acceso.

- Rol de servicio: un rol de servicio es un rol de IAM que adopta un servicio para realizar acciones en su nombre. Un administrador de IAM puede crear, modificar y eliminar un rol de servicio desde IAM. Para obtener más información, consulte Creación de un rol para delegar permisos a un Servicio de AWS en la Guía del usuario de IAM.
- Función vinculada al servicio: una función vinculada a un servicio es un tipo de función de servicio que está vinculada a un. Servicio de AWS El servicio puedes asumir el rol para realizar una acción en su nombre. Los roles vinculados al servicio aparecen en usted Cuenta de AWS y son propiedad del servicio. Un administrador de IAM puedes ver, pero no editar, los permisos de los roles vinculados a servicios.

- Aplicaciones que se ejecutan en Amazon EC2: puedes usar un rol de IAM para administrar las credenciales temporales de las aplicaciones que se ejecutan en una EC2 instancia y realizan AWS CLI solicitudes a la AWS API. Esto es preferible a almacenar las claves de acceso en la EC2 instancia. Para asignar un AWS rol a una EC2 instancia y ponerlo a disposición de todas sus aplicaciones, debe crear un perfil de instancia adjunto a la instancia. Un perfil de instancia contiene el rol y permite que los programas que se ejecutan en la EC2 instancia obtengan credenciales temporales. Para obtener más información, consulte Usar un rol de IAM para conceder permisos a las aplicaciones que se ejecutan en EC2 instancias de Amazon en la Guía del usuario de IAM.

Administración de acceso mediante políticas

El acceso se controla AWS creando políticas y adjuntándolas a AWS identidades o recursos. Una política es un objeto AWS que, cuando se asocia a una identidad o un recurso, define sus permisos. AWS evalúa estas políticas cuando un director (usuario, usuario raíz o sesión de rol) realiza una solicitud. Los permisos en las políticas determinan si la solicitud se permite o se deniega. La mayoría de las políticas se almacenan AWS como documentos JSON. Para obtener más información sobre la estructura y el contenido de los documentos de política JSON, consulta Información general de políticas JSON en la Guía del usuario de IAM.

Los administradores pueden usar las políticas de AWS JSON para especificar quién tiene acceso a qué. Es decir, qué entidad principal puede realizar acciones en qué recursos y en qué condiciones.

De forma predeterminada, los usuarios y los roles no tienen permisos. Un administrador de IAM puedes crear políticas de IAM para conceder permisos a los usuarios para realizar acciones en los recursos que necesitan. A continuación, el administrador puedes añadir las políticas de IAM a roles y los usuarios puedes asumirlos.

Las políticas de IAM definen permisos para una acción independientemente del método que se utiliza para realizar la operación. Por ejemplo, suponga que dispone de una política que permite la acción `iam:GetRole`. Un usuario con esa política puede obtener información sobre el rol de la API AWS Management Console AWS CLI, la o la AWS API.

Políticas basadas en identidades

Las políticas basadas en identidad son documentos de políticas de permisos JSON que puedes asociar a una identidad, como un usuario de IAM, un grupo de usuarios o un rol. Estas políticas controlan qué acciones pueden realizar los usuarios y los roles, en qué recursos y en qué condiciones. Para obtener más información sobre cómo crear una política basada en identidad, consulte Creación de políticas de IAM en la Guía del usuario de IAM.

Las políticas basadas en identidades puedes clasificarse además como políticas insertadas o políticas administradas. Las políticas insertadas se integran directamente en un único usuario, grupo o rol. Las políticas administradas son políticas independientes que puede adjuntar a varios usuarios, grupos y roles de su Cuenta de AWS empresa. Las políticas administradas incluyen políticas AWS administradas y políticas administradas por el cliente. Para obtener más información sobre cómo elegir una política administrada o una política insertada, consulte Elegir entre políticas administradas y políticas insertadas en la Guía del usuario de IAM.

Políticas basadas en recursos

Las políticas basadas en recursos son documentos de política JSON que se asocian a un recurso. Los ejemplos de políticas basadas en recursos son las políticas de confianza de roles de IAM y las políticas de bucket de Amazon S3. En los servicios que admiten políticas basadas en recursos, los administradores de servicios puedes utilizarlos para controlar el acceso a un recurso específico. Para el recurso al que se asocia la política, la política define qué acciones puedes realizar una entidad principal especificada en ese recurso y en qué condiciones. Debe especificar una entidad principal en una política en función de recursos. Los principales pueden incluir cuentas, usuarios, roles, usuarios federados o. Servicios de AWS

Las políticas basadas en recursos son políticas insertadas que se encuentran en ese servicio. No puedes usar políticas AWS gestionadas de IAM en una política basada en recursos.

Listas de control de acceso () ACLs

Las listas de control de acceso (ACLs) controlan qué responsables (miembros de la cuenta, usuarios o roles) tienen permisos para acceder a un recurso. ACLs son similares a las políticas basadas en recursos, aunque no utilizan el formato de documento de políticas JSON.

Amazon S3 y Amazon VPC son ejemplos de servicios compatibles. AWS WAF ACLs Para obtener más información ACLs, consulte la descripción general de la lista de control de acceso (ACL) en la Guía para desarrolladores de Amazon Simple Storage Service.

Otros tipos de políticas

AWS admite tipos de políticas adicionales y menos comunes. Estos tipos de políticas puedes establecer el máximo de permisos que los tipos de políticas más frecuentes le conceden.

- Límites de permisos: un límite de permisos es una característica avanzada que le permite establecer los permisos máximos que una política basada en identidad puedes conceder a una entidad de IAM (usuario o rol de IAM). Puedes establecer un límite de permisos para una entidad. Los permisos resultantes son la intersección de las políticas basadas en la identidad de la entidad y los límites de permisos. Las políticas basadas en recursos que especifiquen el usuario o rol en el campo `Principal` no estarán restringidas por el límite de permisos. Una denegación explícita en cualquiera de estas políticas anulará el permiso. Para obtener más información sobre los límites de los permisos, consulta Límites de permisos para las entidades de IAM en la Guía del usuario de IAM.
- Políticas de control de servicios (SCPs): SCPs son políticas de JSON que especifican los permisos máximos para una organización o unidad organizativa (OU). AWS Organizations AWS Organizations es un servicio para agrupar y gestionar de forma centralizada varios de los Cuentas de AWS que son propiedad de su empresa. Si habilitas todas las funciones de una organización, puedes aplicar políticas de control de servicios (SCPs) a una o a todas tus cuentas. El SCP limita los permisos de las entidades en las cuentas de los miembros, incluidas las de cada una Usuario raíz de la cuenta de AWS. Para obtener más información sobre Organizations SCPs, consulte las políticas de control de servicios en la Guía del AWS Organizations usuario.
- Políticas de control de recursos (RCPs): RCPs son políticas de JSON que puedes usar para establecer los permisos máximos disponibles para los recursos de tus cuentas sin actualizar las políticas de IAM asociadas a cada recurso que poseas. El RCP limita los permisos de los recursos en las cuentas de los miembros y puede afectar a los permisos efectivos de las identidades, incluidos los permisos Usuario raíz de la cuenta de AWS, independientemente de si pertenecen a su organización. Para obtener más información sobre Organizations e RCPs incluir una lista de Servicios de AWS ese apoyo RCPs, consulte Políticas de control de recursos (RCPs) en la Guía del AWS Organizations usuario.
- Políticas de sesión: las políticas de sesión son políticas avanzadas que se pasan como parámetro cuando se crea una sesión temporal mediante programación para un rol o un usuario federado. Los permisos de la sesión resultantes son la intersección de las políticas basadas en identidades

del rol y las políticas de la sesión. Los permisos también puedes proceder de una política en función de recursos. Una denegación explícita en cualquiera de estas políticas anulará el permiso. Para más información, consulta Políticas de sesión en la Guía del usuario de IAM.

Varios tipos de políticas

Cuando se aplican varios tipos de políticas a una solicitud, los permisos resultantes son más complicados de entender. Para saber cómo se AWS determina si se debe permitir una solicitud cuando se trata de varios tipos de políticas, consulte la lógica de evaluación de políticas en la Guía del usuario de IAM.

Cómo funciona Amazon Lookout for Vision con IAM

Antes de utilizar IAM para administrar el acceso a Lookout for Vision, conozca qué características de IAM se pueden utilizar con Lookout for Vision.

Características de IAM que puede utilizar con Amazon Lookout for Vision

Característica de IAM	Asistencia de Lookout for Vision
Políticas basadas en identidades	Sí
Políticas basadas en recursos	No
Acciones de políticas	Sí
Recursos de políticas	Sí
Claves de condición de política (específicas del servicio)	Sí
ACLs	No
ABAC (etiquetas en políticas)	Parcial
Credenciales temporales	Sí
Sesiones de acceso directo (FAS)	Sí

Característica de IAM	Asistencia de Lookout for Vision
Roles de servicio	No
Roles vinculados al servicio	No

Para obtener una visión general de cómo funcionan Lookout for Vision y AWS otros servicios con la mayoría de las funciones de IAM, AWS consulta los servicios que funcionan con IAM en la Guía del usuario de IAM.

Políticas basadas en identidades para Lookout for Vision

Compatibilidad con las políticas basadas en identidad: sí

Las políticas basadas en identidad son documentos de políticas de permisos JSON que puede asociar a una identidad, como un usuario de IAM, un grupo de usuarios o un rol. Estas políticas controlan qué acciones pueden realizar los usuarios y los roles, en qué recursos y en qué condiciones. Para obtener más información sobre cómo crear una política basada en identidad, consulte Creación de políticas de IAM en la Guía del usuario de IAM.

Con las políticas basadas en identidades de IAM, puede especificar las acciones y los recursos permitidos o denegados, así como las condiciones en las que se permiten o deniegan las acciones. No es posible especificar la entidad principal en una política basada en identidad porque se aplica al usuario o rol al que está asociada. Para obtener más información sobre los elementos que puede utilizar en una política de JSON, consulte Referencia de los elementos de las políticas de JSON de IAM en la Guía del usuario de IAM.

Ejemplos de políticas basadas en identidades para Lookout for Vision

Para ver ejemplos de políticas basadas en identidad, consulte Ejemplos de políticas basadas en identidades para Amazon Lookout for Vision.

Políticas basadas en recursos de Lookout for Vision

Admite políticas basadas en recursos: no

Las políticas basadas en recursos son documentos de política JSON que se asocian a un recurso. Los ejemplos de políticas basadas en recursos son las políticas de confianza de roles de IAM y las políticas de bucket de Amazon S3. En los servicios que admiten políticas basadas en recursos, los

administradores de servicios puedes utilizarlos para controlar el acceso a un recurso específico. Para el recurso al que se asocia la política, la política define qué acciones puedes realizar una entidad principal especificada en ese recurso y en qué condiciones. Debe especificar una entidad principal en una política en función de recursos. Los principales pueden incluir cuentas, usuarios, roles, usuarios federados o. Servicios de AWS

Para habilitar el acceso entre cuentas, puede especificar toda una cuenta o entidades de IAM de otra cuenta como la entidad principal de una política en función de recursos. Añadir a una política en función de recursos una entidad principal entre cuentas es solo una parte del establecimiento de una relación de confianza. Cuando el principal y el recurso son diferentes Cuentas de AWS, el administrador de IAM de la cuenta de confianza también debe conceder a la entidad principal (usuario o rol) permiso para acceder al recurso. Para conceder el permiso, adjunte la entidad a una política basada en identidad. Sin embargo, si la política basada en recursos concede acceso a una entidad principal de la misma cuenta, no es necesaria una política basada en identidad adicional. Para obtener más información, consulte Cross account resource access in IAM en la Guía del usuario de IAM.

Acciones de políticas de Lookout for Vision

Compatibilidad con las acciones de políticas: sí

Los administradores pueden usar las políticas de AWS JSON para especificar quién tiene acceso a qué. Es decir, qué entidad principal puedes realizar acciones en qué recursos y en qué condiciones.

El elemento `Action` de una política JSON describe las acciones que puedes utilizar para conceder o denegar el acceso en una política. Las acciones políticas suelen tener el mismo nombre que la operación de AWS API asociada. Hay algunas excepciones, como acciones de solo permiso que no tienen una operación de API coincidente. También hay algunas operaciones que requieren varias acciones en una política. Estas acciones adicionales se denominan acciones dependientes.

Incluya acciones en una política para conceder permisos y así llevar a cabo la operación asociada.

Para ver una lista de acciones de Lookout for Vision, consulte Acciones definidas por Amazon Lookout for Vision en la Referencia de autorizaciones de servicio.

Las acciones de políticas de Lookout for Vision utilizan el siguiente prefijo antes de la acción:

```
lookoutvision
```

Para especificar varias acciones en una única instrucción, sepárelas con comas.

```
"Action": [
      "lookoutvision:action1",
      "lookoutvision:action2"
         ]
```

Para ver ejemplos de políticas basadas en identidad, consulte Ejemplos de políticas basadas en identidades para Amazon Lookout for Vision.

Recursos de políticas de Lookout for Vision

Compatibilidad con los recursos de políticas: sí

Los administradores pueden usar las políticas de AWS JSON para especificar quién tiene acceso a qué. Es decir, qué entidad principal puedes realizar acciones en qué recursos y en qué condiciones.

El elemento `Resource` de la política JSON especifica el objeto u objetos a los que se aplica la acción. Las instrucciones deben contener un elemento `Resource` o `NotResource`. Como práctica recomendada, especifique un recurso utilizando el Nombre de recurso de Amazon (ARN). Puedes hacerlo para acciones que admitan un tipo de recurso específico, conocido como permisos de nivel de recurso.

Para las acciones que no admiten permisos de nivel de recurso, como las operaciones de descripción, utiliza un carácter comodín (*) para indicar que la instrucción se aplica a todos los recursos.

```
"Resource": "*"
```

Para ver una lista de los tipos de recursos de Lookout for Vision y ARNs sus tipos de recursos, consulte Recursos definidos por Amazon Lookout for Vision en la Referencia de autorización de servicios. Para obtener información sobre las acciones con las que puede especificar el ARN de cada recurso, consulte Acciones definidas por Amazon Lookout for Vision.

Para ver ejemplos de políticas basadas en identidad, consulte Ejemplos de políticas basadas en identidades para Amazon Lookout for Vision.

Claves de condición de política de Amazon Lookout for Vision

Compatibilidad con claves de condición de políticas específicas del servicio: sí

Los administradores pueden usar las políticas de AWS JSON para especificar quién tiene acceso a qué. Es decir, qué entidad principal puedes realizar acciones en qué recursos y en qué condiciones.

El elemento `Condition` (o bloque de `Condition`) permite especificar condiciones en las que entra en vigor una instrucción. El elemento `Condition` es opcional. Puedes crear expresiones condicionales que utilizan operadores de condición, tales como igual o menor que, para que la condición de la política coincida con los valores de la solicitud.

Si especifica varios elementos de `Condition` en una instrucción o varias claves en un único elemento de `Condition`, AWS las evalúa mediante una operación `AND` lógica. Si especifica varios valores para una única clave de condición, AWS evalúa la condición mediante una `OR` operación lógica. Se deben cumplir todas las condiciones antes de que se concedan los permisos de la instrucción.

También puedes utilizar variables de marcador de posición al especificar condiciones. Por ejemplo, puedes conceder un permiso de usuario de IAM para acceder a un recurso solo si está etiquetado con su nombre de usuario de IAM. Para más información, consulta Elementos de la política de IAM: variables y etiquetas en la Guía del usuario de IAM.

AWS admite claves de condición globales y claves de condición específicas del servicio. Para ver todas las claves de condición AWS globales, consulte las claves de contexto de condición AWS globales en la Guía del usuario de IAM.

Para ver una lista de las claves de condición de Lookout for Vision, consulte Claves de condición para Amazon Lookout for Vision en la Referencia de autorizaciones de servicio. Para obtener más información acerca de las acciones y los recursos con los que puede utilizar una clave de condición, consulte Acciones definidas por Amazon Lookout for Vision.

Para ver ejemplos de políticas basadas en identidad, consulte Ejemplos de políticas basadas en identidades para Amazon Lookout for Vision.

ACLs en Lookout for Vision

Soportes ACLs: No

Las listas de control de acceso (ACLs) controlan qué directores (miembros de la cuenta, usuarios o roles) tienen permisos para acceder a un recurso. ACLs son similares a las políticas basadas en recursos, aunque no utilizan el formato de documento de políticas JSON.

ABAC con Lookout for Vision

Compatibilidad con ABAC (etiquetas en las políticas): parcial

El control de acceso basado en atributos (ABAC) es una estrategia de autorización que define permisos en función de atributos. En AWS, estos atributos se denominan etiquetas. Puede adjuntar etiquetas a las entidades de IAM (usuarios o roles) y a muchos AWS recursos. El etiquetado de entidades y recursos es el primer paso de ABAC. A continuación, designa las políticas de ABAC para permitir operaciones cuando la etiqueta de la entidad principal coincida con la etiqueta del recurso al que se intenta acceder.

ABAC es útil en entornos que crecen con rapidez y ayuda en situaciones en las que la administración de las políticas resulta engorrosa.

Para controlar el acceso en función de etiquetas, debe proporcionar información de las etiquetas en el elemento de condición de una política utilizando las claves de condición `aws:ResourceTag/key-name`, `aws:RequestTag/key-name` o `aws:TagKeys`.

Si un servicio admite las tres claves de condición para cada tipo de recurso, el valor es Sí para el servicio. Si un servicio admite las tres claves de condición solo para algunos tipos de recursos, el valor es Parcial.

Para obtener más información sobre ABAC, consulte Definición de permisos con la autorización de ABAC en la Guía del usuario de IAM. Para ver un tutorial con los pasos para configurar ABAC, consulta Uso del control de acceso basado en atributos (ABAC) en la Guía del usuario de IAM.

Uso de credenciales temporales con Lookout for Vision

Compatibilidad con credenciales temporales: sí

Algunas Servicios de AWS no funcionan cuando inicias sesión con credenciales temporales. Para obtener información adicional, incluida información sobre cuáles Servicios de AWS funcionan con credenciales temporales, consulta Cómo Servicios de AWS funcionan con IAM en la Guía del usuario de IAM.

Utiliza credenciales temporales si inicia sesión en ellas AWS Management Console mediante cualquier método excepto un nombre de usuario y una contraseña. Por ejemplo, cuando accedes

AWS mediante el enlace de inicio de sesión único (SSO) de tu empresa, ese proceso crea automáticamente credenciales temporales. También crea credenciales temporales de forma automática cuando inicia sesión en la consola como usuario y luego cambia de rol. Para obtener más información sobre el cambio de roles, consulte Cambio de un usuario a un rol de IAM (consola) en la Guía del usuario de IAM.

Puedes crear credenciales temporales manualmente mediante la AWS CLI API o. AWS A continuación, puede utilizar esas credenciales temporales para acceder AWS. AWS recomienda generar credenciales temporales de forma dinámica en lugar de utilizar claves de acceso a largo plazo. Para obtener más información, consulte Credenciales de seguridad temporales en IAM.

Sesiones de acceso directo para Lookout for Vision

Admite sesiones de acceso directo (FAS): sí

Cuando utilizas un usuario o un rol de IAM para realizar acciones en AWSél, se te considera director. Cuando utiliza algunos servicios, es posible que realice una acción que desencadene otra acción en un servicio diferente. FAS utiliza los permisos del principal que llama y los que solicita Servicio de AWS para realizar solicitudes a los servicios descendentes. Servicio de AWS Las solicitudes de FAS solo se realizan cuando un servicio recibe una solicitud que requiere interacciones con otros Servicios de AWS recursos para completarse. En este caso, debe tener permisos para realizar ambas acciones. Para obtener información sobre las políticas a la hora de realizar solicitudes de FAS, consulta Reenviar sesiones de acceso.

Roles de servicio de Lookout for Vision

Compatible con roles de servicio: No

Un rol de servicio es un rol de IAM que asume un servicio para realizar acciones en su nombre. Un administrador de IAM puede crear, modificar y eliminar un rol de servicio desde IAM. Para obtener más información, consulte Creación de un rol para delegar permisos a un Servicio de AWS en la Guía del usuario de IAM.

> **Warning**
>
> Cambiar los permisos de un rol de servicio podría interrumpir la funcionalidad de Lookout for Vision. Edite los roles de servicio solo cuando Lookout for Vision proporcione orientación para hacerlo.

Roles vinculados a servicios de Lookout for Vision

Compatibilidad con roles vinculados al servicio: no

Un rol vinculado a un servicio es un tipo de rol de servicio que está vinculado a un. Servicio de AWS El servicio puedes asumir el rol para realizar una acción en su nombre. Los roles vinculados al servicio aparecen en usted Cuenta de AWS y son propiedad del servicio. Un administrador de IAM puedes ver, pero no editar, los permisos de los roles vinculados a servicios.

Para más información sobre cómo crear o administrar roles vinculados a servicios, consulta Servicios de AWS que funcionan con IAM. Busque un servicio en la tabla que incluya Yes en la columna Rol vinculado a un servicio. Seleccione el vínculo Sí para ver la documentación acerca del rol vinculado a servicios para ese servicio.

Ejemplos de políticas basadas en identidades para Amazon Lookout for Vision

De forma predeterminada, los usuarios y roles no tienen permiso para crear, ver ni modificar recursos de Lookout for Vision. Tampoco pueden realizar tareas mediante la AWS Management Console, AWS Command Line Interface (AWS CLI) o AWS la API. Un administrador de IAM puedes crear políticas de IAM para conceder permisos a los usuarios para realizar acciones en los recursos que necesitan. A continuación, el administrador puedes añadir las políticas de IAM a roles y los usuarios puedes asumirlos.

Para obtener información acerca de cómo crear una política basada en identidades de IAM mediante el uso de estos documentos de políticas JSON de ejemplo, consulte Creación de políticas de IAM (consola) en la Guía del usuario de IAM.

Para obtener más información sobre las acciones y los tipos de recursos definidos por Lookout for Vision, incluido el formato de ARNs cada uno de los tipos de recursos, consulte Acciones, recursos y claves de condición de Amazon Lookout for Vision en la Referencia de autorización de servicio.

Temas

- Prácticas recomendadas sobre las políticas
- Acceso a un único proyecto de Amazon Lookout for Vision
- Ejemplo de política basada en etiquetas

Prácticas recomendadas sobre las políticas

Las políticas basadas en identidades determinan si alguien puede crear, acceder o eliminar los recursos de Lookout for Vision de la cuenta. Estas acciones pueden generar costos adicionales para su Cuenta de AWS. Siga estas directrices y recomendaciones al crear o editar políticas basadas en identidades:

- Comience con las políticas AWS administradas y avance hacia los permisos con privilegios mínimos: para empezar a conceder permisos a sus usuarios y cargas de trabajo, utilice las políticas AWS administradas que otorgan permisos en muchos casos de uso comunes. Están disponibles en su. Cuenta de AWS Le recomendamos que reduzca aún más los permisos definiendo políticas administradas por el AWS cliente que sean específicas para sus casos de uso. Con el fin de obtener más información, consulta las políticas administradas por AWS o las políticas administradas por AWS para funciones de tarea en la Guía de usuario de IAM.
- Aplique permisos de privilegio mínimo: cuando establezca permisos con políticas de IAM, conceda solo los permisos necesarios para realizar una tarea. Para ello, debe definir las acciones que se puedes llevar a cabo en determinados recursos en condiciones específicas, también conocidos como permisos de privilegios mínimos. Con el fin de obtener más información sobre el uso de IAM para aplicar permisos, consulta Políticas y permisos en IAM en la Guía del usuario de IAM.
- Utiliza condiciones en las políticas de IAM para restringir aún más el acceso: puedes agregar una condición a sus políticas para limitar el acceso a las acciones y los recursos. Por ejemplo, puedes escribir una condición de políticas para especificar que todas las solicitudes deben enviarse utilizando SSL. También puedes usar condiciones para conceder el acceso a las acciones del servicio si se utilizan a través de una acción específica Servicio de AWS, por ejemplo AWS CloudFormation. Para obtener más información, consulta Elementos de la política de JSON de IAM: Condición en la Guía del usuario de IAM.
- Utiliza el analizador de acceso de IAM para validar las políticas de IAM con el fin de garantizar la seguridad y funcionalidad de los permisos: el analizador de acceso de IAM valida políticas nuevas y existentes para que respeten el lenguaje (JSON) de las políticas de IAM y las prácticas recomendadas de IAM. El analizador de acceso de IAM proporciona más de 100 verificaciones de políticas y recomendaciones procesables para ayudar a crear políticas seguras y funcionales. Para más información, consulte Validación de políticas con el Analizador de acceso de IAM en la Guía del usuario de IAM.
- Requerir autenticación multifactor (MFA): si tiene un escenario que requiere usuarios de IAM o un usuario raíz en Cuenta de AWS su cuenta, active la MFA para mayor seguridad. Para exigir la

MFA cuando se invoquen las operaciones de la API, añada condiciones de MFA a sus políticas. Para más información, consulte Acceso seguro a la API con MFA en la Guía del usuario de IAM.

Para obtener más información sobre las prácticas recomendadas de IAM, consulte Prácticas recomendadas de seguridad en IAM en la Guía del usuario de IAM.

Acceso a un único proyecto de Amazon Lookout for Vision

En este ejemplo, quieres conceder a un usuario de tu AWS cuenta acceso a uno de tus proyectos de Amazon Lookout for Vision.

```
{
    "Sid": "SpecificProjectOnly",
    "Effect": "Allow",
    "Action": [
        "lookoutvision:DetectAnomalies"
    ],
    "Resource": "arn:aws:lookoutvision:us-east-1:123456789012:model/myproject/*"
}
```

Ejemplo de política basada en etiquetas

Las políticas basadas en etiquetas son documentos de política JSON que especifican las acciones que una entidad principal puede realizar en recursos etiquetados.

Usar una etiqueta para acceder a un recurso

Esta política de ejemplo concede permiso a un usuario o rol de su cuenta de AWS para utilizar la operación `DetectAnomalies` con cualquier recurso etiquetado con la clave `stage` y el valor `production`.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "LookoutVision:DetectAnomalies"
            ],
            "Resource": "*",
```

```
            "Condition": {
                "StringEquals": {
                    "aws:ResourceTag/stage": "production"
                }
            }
        }
    ]
}
```

Use una etiqueta para denegar el acceso a operaciones específicas de Amazon Lookout for Vision

Esta política de ejemplo niega el permiso a un usuario o rol de su cuenta de AWS a llamar operaciones `DeleteModel` o `StopModel` con cualquier modelo etiquetado con la clave `stage` y el valor `production`.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": [
                "LookoutVision:DeleteModel",
                "LookoutVision:StopModel"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:ResourceTag/stage": "production"
                }
            }
        }
    ]
}
```

AWS políticas gestionadas para Amazon Lookout for Vision

Una política AWS administrada es una política independiente creada y administrada por. AWS AWS Las políticas administradas están diseñadas para proporcionar permisos para muchos casos de uso comunes, de modo que pueda empezar a asignar permisos a usuarios, grupos y funciones.

Ten en cuenta que es posible que las políticas AWS administradas no otorguen permisos con privilegios mínimos para tus casos de uso específicos, ya que están disponibles para que los usen todos los AWS clientes. Se recomienda definir políticas administradas por el cliente específicas para sus casos de uso a fin de reducir aún más los permisos.

No puedes cambiar los permisos definidos en AWS las políticas administradas. Si AWS actualiza los permisos definidos en una política AWS administrada, la actualización afecta a todas las identidades principales (usuarios, grupos y roles) a las que está asociada la política. AWS es más probable que actualice una política AWS administrada cuando Servicio de AWS se lance una nueva o cuando estén disponibles nuevas operaciones de API para los servicios existentes.

Para obtener más información, consulte Políticas administradas de AWS en la Guía del usuario de IAM.

Política gestionada por AWS: AmazonLookoutVisionReadOnlyAccess

Utilice la `AmazonLookoutVisionReadOnlyAccess` política para permitir a los usuarios el acceso de solo lectura a Amazon Lookout for Vision (y sus dependencias) con las siguientes acciones de Amazon Lookout for Vision (operaciones del SDK). Por ejemplo, se puede utilizar `DescribeModel` para obtener información sobre un modelo existente.

- DescribeDataset
- DescribeModel
- DescribeModelPackagingJob
- DescribeProject
- ListDatasetEntries
- ListModelPackagingJobs
- ListModels
- ListProjects
- ListTagsForResource

Para llamar acciones de solo lectura, los usuarios no necesitan permisos de bucket de Amazon S3. Sin embargo, las respuestas de las operaciones pueden incluir referencias a buckets de Amazon S3. Por ejemplo, la entrada `source-ref` en la respuesta de `ListDatasetEntries` es una

referencia a una imagen de un bucket de Amazon S3. Añada permisos de bucket de Amazon S3 si sus usuarios necesitan acceder a los buckets referenciados. Por ejemplo, es posible que un usuario desee descargar una imagen a la que hace referencia un campo `source-ref`. Para obtener más información, consulte Concesión de permisos de bucket de Amazon S3.

Puede adjuntar la política de `AmazonLookoutVisionReadOnlyAccess` a las identidades de IAM.

Detalles de los permisos

Esta política incluye los siguientes permisos.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "LookoutVisionReadOnlyAccess",
            "Effect": "Allow",
            "Action": [
                "lookoutvision:DescribeDataset",
                "lookoutvision:DescribeModel",
                "lookoutvision:DescribeProject",
                "lookoutvision:DescribeModelPackagingJob",
                "lookoutvision:ListDatasetEntries",
                "lookoutvision:ListModels",
                "lookoutvision:ListProjects",
                "lookoutvision:ListTagsForResource",
                "lookoutvision:ListModelPackagingJobs"
            ],
            "Resource": "*"
        }
    ]
}
```

Política gestionada por AWS: AmazonLookoutVisionFullAccess

Utilice la `AmazonLookoutVisionFullAccess` política para permitir a los usuarios el acceso de acceso completo a Amazon Lookout for Vision (y sus dependencias) con acciones de Amazon Lookout for Vision (operaciones del SDK). Por ejemplo, puede entrenar un modelo sin tener que usar la consola de Amazon Lookout for Vision. Para obtener más información, consulte Acciones.

Para crear un conjunto de datos (`CreateDataset`) o crear un modelo (`CreateModel`), sus usuarios deben tener permisos de acceso total al depósito de Amazon S3 que almacena las imágenes del conjunto de datos, los archivos de manifiesto de Amazon SageMaker AI Ground Truth y los resultados de la formación. Para obtener más información, consulte Paso 2: configuración de permisos.

También puede conceder permiso a las acciones del SDK de Amazon Lookout for Vision mediante la política `AmazonLookoutVisionConsoleFullAccess`.

Puede adjuntar la política de `AmazonLookoutVisionFullAccess` a las identidades de IAM.

Detalles de los permisos

Esta política incluye los siguientes permisos.

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

Política gestionada por AWS: AmazonLookoutVisionConsoleFullAccess

Utilice la `AmazonLookoutVisionFullAccess` política para permitir a los usuarios un acceso total a la consola de Amazon Lookout for Vision, a las acciones (operaciones del SDK) y a cualquier dependencia que tenga el servicio. Para obtener más información, consulte Introducción a Amazon Lookout for Vision.

La política `LookoutVisionConsoleFullAccess` incluye los permisos para el bucket de consola de Amazon Lookout for Vision. Para obtener información sobre cómo usar el bucket de la consola, consulte Paso 3: Crear un bucket de consola. Para almacenar conjuntos de datos, imágenes

y archivos de manifiesto de Amazon SageMaker AI Ground Truth en un bucket de Amazon S3 diferente, los usuarios necesitan permisos adicionales. Para obtener más información, consulte the section called “Ajuste de permisos de bucket de Amazon S3”.

Puede adjuntar la política `AmazonLookoutVisionConsoleFullAccess` a las identidades de IAM.

Agrupaciones de permisos

Esta política se agrupa en instrucciones basadas en el conjunto de permisos proporcionados:

- `LookoutVisionFullAccess`: permite el acceso para realizar todas las acciones de Lookout for Vision.
- `LookoutVisionConsoleS3BucketSearchAccess`: permite enumerar todos los buckets de Amazon S3 propiedad de la persona que llama. Lookout for Vision usa esta acción para identificar el bucket de consola de Lookout for Vision específico de la región de AWS, si existe alguno en la cuenta de la persona que llama.
- `LookoutVisionConsoleS3BucketFirstUseSetupAccessPermissions`: permite crear y configurar buckets de Amazon S3 que coincidan con el patrón del nombre del bucket de la consola de Lookout for Vision. Lookout for Vision utiliza estas acciones para crear y configurar un bucket de consola de Lookout for Vision específico de una región cuando no lo encuentra.
- `LookoutVisionConsoleS3BucketAccess`: permite acciones dependientes de Amazon S3 en buckets que coinciden con el patrón del nombre del bucket de la consola de Lookout for Vision. Lookout for Vision utiliza `s3:ListBucket` para buscar objetos de imagen al crear un conjunto de datos a partir de un bucket de Amazon S3 y al iniciar una tarea de prueba de detección. Lookout for Vision `s3:GetBucketLocation` utiliza `s3:GetBucketVersioning` y valida la región, el propietario y la configuración AWS del bucket como parte de lo siguiente:
  - Creación de un conjunto de datos
  - Entrenamiento de un modelo
  - Inicio de una tarea de prueba de detección
  - Comentarios sobre las pruebas de detección

  `LookoutVisionConsoleS3ObjectAccess`: permite leer y escribir objetos de Amazon S3 dentro de buckets que coinciden con el patrón de nombres del bucket de la consola de Lookout for Vision. Lookout for Vision usa estas acciones para mostrar imágenes en las vistas de galería de la consola y para cargar nuevas imágenes para usarlas en conjuntos de datos. Además, estos permisos

permiten a Lookout for Vision escribir metadatos mientras crea un conjunto de datos, entrena un modelo, inicia una tarea de pruebas de detección y realiza comentarios sobre las pruebas de detección.

- `LookoutVisionConsoleDatasetLabelingToolsAccess`— Permite acciones de GroundTruth etiquetado dependientes de Amazon SageMaker AI. Lookout for Vision utiliza estas acciones para escanear depósitos de S3 en busca de imágenes, GroundTruth crear archivos de manifiesto y anotar los resultados de las tareas de detección de ensayos con etiquetas de validación.
- `LookoutVisionConsoleDashboardAccess`- Permite leer las CloudWatch métricas de Amazon. Lookout for Vision utiliza estas acciones para rellenar los gráficos del panel de control y las estadísticas de anomalías detectadas.
- `LookoutVisionConsoleTagSelectorAccess`: permite leer sugerencias de claves y valores de etiquetas específicas de la cuenta. Lookout for Vision usa estos permisos para ofrecer recomendaciones de claves y valores de etiquetas en las páginas de la consola Administrar etiquetas.
- `LookoutVisionConsoleKmsKeySelectorAccess`— Permite enumerar claves y alias AWS Key Management Service (KMS). Amazon Lookout for Vision utiliza este permiso para rellenar las claves de KMS en la selección de Etiquetas sugerida en determinadas acciones de Lookout for Vision que admiten claves de KMS administradas por el cliente para el cifrado.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "LookoutVisionFullAccess",
            "Effect": "Allow",
            "Action": [
                "lookoutvision:*"
            ],
            "Resource": "*"
        },
        {
            "Sid": "LookoutVisionConsoleS3BucketSearchAccess",
            "Effect": "Allow",
            "Action": [
                "s3:ListAllMyBuckets"
            ],
            "Resource": "*"
        },
```

```
        {
            "Sid": "LookoutVisionConsoleS3BucketFirstUseSetupAccess",
            "Effect": "Allow",
            "Action": [
                "s3:CreateBucket",
                "s3:PutBucketVersioning",
                "s3:PutLifecycleConfiguration",
                "s3:PutEncryptionConfiguration",
                "s3:PutBucketPublicAccessBlock"
            ],
            "Resource": "arn:aws:s3:::lookoutvision-*"
        },
        {
            "Sid": "LookoutVisionConsoleS3BucketAccess",
            "Effect": "Allow",
            "Action": [
                "s3:ListBucket",
                "s3:GetBucketLocation",
                "s3:GetBucketAcl",
                "s3:GetBucketVersioning"
            ],
            "Resource": "arn:aws:s3:::lookoutvision-*"
        },
        {
            "Sid": "LookoutVisionConsoleS3ObjectAccess",
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion",
                "s3:PutObject",
                "s3:AbortMultipartUpload",
                "s3:ListMultipartUploadParts"
            ],
            "Resource": "arn:aws:s3:::lookoutvision-*/*"
        },
        {
            "Sid": "LookoutVisionConsoleDatasetLabelingToolsAccess",
            "Effect": "Allow",
            "Action": [
                "groundtruthlabeling:RunGenerateManifestByCrawlingJob",
                "groundtruthlabeling:AssociatePatchToManifestJob",
                "groundtruthlabeling:DescribeConsoleJob"
            ],
            "Resource": "*"
```

```
        },
        {
            "Sid": "LookoutVisionConsoleDashboardAccess",
            "Effect": "Allow",
            "Action": [
                "cloudwatch:GetMetricData",
                "cloudwatch:GetMetricStatistics"
            ],
            "Resource": "*"
        },
        {
            "Sid": "LookoutVisionConsoleTagSelectorAccess",
            "Effect": "Allow",
            "Action": [
                "tag:GetTagKeys",
                "tag:GetTagValues"
            ],
            "Resource": "*"
        },
        {
            "Sid": "LookoutVisionConsoleKmsKeySelectorAccess",
            "Effect": "Allow",
            "Action": [
                "kms:ListAliases"
            ],
            "Resource": "*"
        }
    ]
}
```

Política gestionada por AWS: AmazonLookoutVisionConsoleReadOnlyAccess

Utilice la política `AmazonLookoutVisionConsoleReadOnlyAccess` para permitir a los usuarios un acceso total a la consola de Amazon Lookout for Vision, a las acciones (operaciones del SDK) y a cualquier dependencia que tenga el servicio.

La política `AmazonLookoutVisionConsoleReadOnlyAccess` incluye los permisos de Amazon S3 para el bucket de consola de Amazon Lookout for Vision. Si las imágenes de su conjunto de datos o los archivos de manifiesto de Amazon SageMaker AI Ground Truth se encuentran en un bucket de Amazon S3 diferente, sus usuarios necesitan permisos adicionales. Para obtener más información, consulte the section called “Ajuste de permisos de bucket de Amazon S3”.

Puede adjuntar la política `AmazonLookoutVisionConsoleReadOnlyAccess` a las identidades de IAM.

Agrupaciones de permisos

Esta política se agrupa en instrucciones basadas en el conjunto de permisos proporcionados:

- `LookoutVisionReadOnlyAccess`: permite el acceso para realizar acciones de Lookout for Vision de solo lectura.
- `LookoutVisionConsoleS3BucketSearchAccess`: permite enumerar todos los buckets de Amazon S3 propiedad de la persona que llama. Lookout for Vision usa esta acción para identificar el bucket de la consola de Lookout for Vision específico de la región de AWS, si existe alguno en la cuenta de la persona que llama.
- `LookoutVisionConsoleS3ObjectReadAccess`: permite leer objetos de Amazon S3 y versiones de objetos de Amazon S3 en los buckets de consola de Lookout for Vision. Lookout for Vision utiliza estas acciones para mostrar las imágenes en conjuntos de datos, modelos y pruebas de detección.
- `LookoutVisionConsoleDashboardAccess`— Permite leer CloudWatch las métricas de Amazon. Lookout for Vision utiliza estas acciones para rellenar los gráficos del panel de control y las estadísticas de anomalías detectadas.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "LookoutVisionReadOnlyAccess",
            "Effect": "Allow",
            "Action": [
                "lookoutvision:DescribeDataset",
                "lookoutvision:DescribeModel",
                "lookoutvision:DescribeProject",
                "lookoutvision:DescribeTrialDetection",
                "lookoutvision:DescribeModelPackagingJob",
                "lookoutvision:ListDatasetEntries",
                "lookoutvision:ListModels",
                "lookoutvision:ListProjects",
                "lookoutvision:ListTagsForResource",
                "lookoutvision:ListTrialDetections",
```

```
                "lookoutvision:ListModelPackagingJobs"
            ],
            "Resource": "*"
        },
        {
            "Sid": "LookoutVisionConsoleS3BucketSearchAccess",
            "Effect": "Allow",
            "Action": [
                "s3:ListAllMyBuckets"
            ],
            "Resource": "*"
        },
        {
            "Sid": "LookoutVisionConsoleS3ObjectReadAccess",
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion"
            ],
            "Resource": "arn:aws:s3:::lookoutvision-*/*"
        },
        {
            "Sid": "LookoutVisionConsoleDashboardAccess",
            "Effect": "Allow",
            "Action": [
                "cloudwatch:GetMetricData",
                "cloudwatch:GetMetricStatistics"
            ],
            "Resource": "*"
        }
    ]
}
```

Busque actualizaciones AWS de las políticas gestionadas de Lookout for Vision

Consulta los detalles sobre las actualizaciones de las políticas AWS gestionadas de Lookout for Vision desde que este servicio comenzó a rastrear estos cambios. Para obtener alertas automáticas sobre cambios en esta página, suscríbase a la fuente RSS en la página de historial de documentos de Amazon Lookout for Vision.

empaquetado de modelos	agregó los siguientes modelos de operaciones de empaquetado a AmazonLookoutVisionFullAccesslas políticas AmazonLookoutVisionConsoleFullAccessand: • DescribeModelPackagingJob • ListModelPackagingJobs • StartModelPackagingJob Amazon Lookout for Vision agregó los siguientes modelos de operaciones de empaquetado a AmazonLookoutVisionReadOnlyAccesslas políticas AmazonLookoutVisionConsoleReadOnlyAccessand: • DescribeModelPackagingJob • ListModelPackagingJobs	
Nuevas políticas añadidas	Amazon Lookout for Vision agregó las siguientes políticas. • AmazonLookoutVisionReadOnlyAccess • AmazonLookoutVisionFullAccess • AmazonLookoutVisionConsoleFullAccess • AmazonLookoutVisionConsoleReadOnlyAccess	11 de mayo de 2021
Lookout for Vision comenzó a realizar el seguimiento de los cambios	Amazon Lookout for Vision comenzó a realizar un seguimiento de los cambios	1 de marzo de 2021

Cambio	Descripción	Fecha
Se agregaron operaciones de empaquetado de modelos	Amazon Lookout for Vision agregó los siguientes modelos de operaciones de empaquetado a AmazonLookoutVisionFullAccesslas políticas AmazonLookoutVisionConsoleFullAccessand: • DescribeModelPackagingJob • ListModelPackagingJobs • StartModelPackagingJob Amazon Lookout for Vision agregó los siguientes modelos de operaciones de empaquetado a AmazonLookoutVisionReadOnlyAccesslas políticas AmazonLookoutVisionConsoleReadOnlyAccessand: • DescribeModelPackagingJob • ListModelPackagingJobs	7 de diciembre de 2021
Nuevas políticas añadidas	Amazon Lookout for Vision agregó las siguientes políticas. • AmazonLookoutVisionReadOnlyAccess • AmazonLookoutVisionFullAccess • AmazonLookoutVisionConsoleFullAccess • AmazonLookoutVisionConsoleReadOnlyAccess en AWS sus políticas gestionadas.	11 de mayo de 2021

Solución de problemas de identidad y acceso de Amazon Lookout for Vision

Utilice la siguiente información para diagnosticar y solucionar los problemas comunes que puedan surgir cuando trabaje con Lookout for Vision e IAM.

Temas

No tengo autorización para realizar una acción en Amazon Lookout for Vision

Si recibe un error que indica que no tiene autorización para realizar una acción, las políticas se deben actualizar para permitirle realizar la acción.

En el siguiente ejemplo, el error se produce cuando el usuario de IAM `mateojackson` intenta utilizar la consola para consultar los detalles acerca de un recurso ficticio *`my-example-widget`*, pero no tiene los permisos ficticios `lookoutvision:`*`GetWidget`*.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 lookoutvision:GetWidget on resource: my-example-widget
```

En este caso, la política del usuario `mateojackson` debe actualizarse para permitir el acceso al recurso *`my-example-widget`* mediante la acción `lookoutvision:`*`GetWidget`*.

Si necesitas ayuda, ponte en contacto con tu AWS administrador. El gestionador es la persona que le proporcionó las credenciales de inicio de sesión.

No estoy autorizado a realizar tareas como: PassRole

Si recibe un error que indica que no tiene autorización para realizar la acción `iam:PassRole`, se deben actualizar las políticas a fin de permitirle pasar un rol a Lookout for Vision.

Algunas Servicios de AWS permiten transferir una función existente a ese servicio en lugar de crear una nueva función de servicio o una función vinculada a un servicio. Para ello, debe tener permisos para transferir el rol al servicio.

En el siguiente ejemplo, el error se produce cuando un usuario de IAM denominado `marymajor` intenta utilizar la consola para realizar una acción en Lookout for Vision. Sin embargo, la acción requiere que el servicio cuente con permisos que otorguen un rol de servicio. Mary no tiene permisos para transferir el rol al servicio.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

En este caso, las políticas de Mary se deben actualizar para permitirle realizar la acción `iam:PassRole`.

Si necesita ayuda, póngase en contacto con su administrador. AWS El gestionador es la persona que le proporcionó las credenciales de inicio de sesión.

Quiero permitir que personas ajenas a mí accedan Cuenta de AWS a mis recursos de Lookout for Vision

Puedes crear un rol que los usuarios de otras cuentas o las personas externas a la organización puedan utilizar para acceder a sus recursos. Puedes especificar una persona de confianza para que asuma el rol. En el caso de los servicios que respaldan las políticas basadas en recursos o las listas de control de acceso (ACLs), puedes usar esas políticas para permitir que las personas accedan a tus recursos.

Para obtener más información, consulte lo siguiente:

- Para obtener información acerca de si Lookout for Vision admite estas características, consulte Cómo funciona Amazon Lookout for Vision con IAM.
- Para obtener información sobre cómo proporcionar acceso a los recursos de su Cuentas de AWS propiedad, consulte Proporcionar acceso a un usuario de IAM en otro de su propiedad en la Cuenta de AWS Guía del usuario de IAM.
- Para obtener información sobre cómo proporcionar acceso a tus recursos a terceros Cuentas de AWS, consulta Cómo proporcionar acceso a recursos que Cuentas de AWS son propiedad de terceros en la Guía del usuario de IAM.
- Para obtener información sobre cómo proporcionar acceso mediante una federación de identidades, consulta Proporcionar acceso a usuarios autenticados externamente (identidad federada) en la Guía del usuario de IAM.
- Para conocer sobre la diferencia entre las políticas basadas en roles y en recursos para el acceso entre cuentas, consulte Acceso a recursos entre cuentas en IAM en la Guía del usuario de IAM.

Validación de la conformidad para Amazon Lookout for Vision

Los auditores externos evalúan la seguridad y la conformidad de Amazon Lookout for Vision como parte de AWS varios programas de conformidad. Amazon Lookout for Vision cumple con el Reglamento General de Protección de Datos (General Data Protection Regulation).

Para saber si un programa de cumplimiento Servicio de AWS está dentro del ámbito de aplicación de programas de cumplimiento específicos, consulte Servicios de AWS Alcance por programa de cumplimiento Servicios de AWS y elija el programa de cumplimiento que le interese. Para obtener información general, consulte Programas de AWS cumplimiento > Programas AWS .

Puede descargar informes de auditoría de terceros utilizando AWS Artifact. Para obtener más información, consulte Descarga de informes en AWS Artifact .

Su responsabilidad de cumplimiento al Servicios de AWS utilizarlos viene determinada por la confidencialidad de sus datos, los objetivos de cumplimiento de su empresa y las leyes y reglamentos aplicables. AWS proporciona los siguientes recursos para ayudar con el cumplimiento:

- Cumplimiento de seguridad y gobernanza: en estas guías se explican las consideraciones de arquitectura y se proporcionan pasos para implementar las características de seguridad y cumplimiento.
- Referencia de servicios válidos de HIPAA: muestra una lista con los servicios válidos de HIPAA. No todos Servicios de AWS cumplen con los requisitos de la HIPAA.
- AWS Recursos de de cumplimiento: esta colección de libros de trabajo y guías puede aplicarse a su industria y ubicación.
- AWS Guías de cumplimiento para clientes: comprenda el modelo de responsabilidad compartida desde el punto de vista del cumplimiento. Las guías resumen las mejores prácticas para garantizar la seguridad Servicios de AWS y orientan los controles de seguridad en varios marcos (incluidos el Instituto Nacional de Estándares y Tecnología (NIST), el Consejo de Normas de Seguridad del Sector de Tarjetas de Pago (PCI) y la Organización Internacional de Normalización (ISO)).
- Evaluación de los recursos con reglas en la guía para AWS Config desarrolladores: el AWS Config servicio evalúa en qué medida las configuraciones de los recursos cumplen con las prácticas internas, las directrices del sector y las normas.
- AWS Security Hub— Esto Servicio de AWS proporciona una visión completa del estado de su seguridad interior AWS. Security Hub utiliza controles de seguridad para evaluar sus recursos de AWS y comprobar su cumplimiento con los estándares y las prácticas recomendadas del

sector de la seguridad. Para obtener una lista de los servicios y controles compatibles, consulta la Referencia de controles de Security Hub.

- Amazon GuardDuty: Servicio de AWS detecta posibles amenazas para sus cargas de trabajo Cuentas de AWS, contenedores y datos mediante la supervisión de su entorno para detectar actividades sospechosas y maliciosas. GuardDuty puede ayudarlo a cumplir con varios requisitos de conformidad, como el PCI DSS, al cumplir con los requisitos de detección de intrusiones exigidos por ciertos marcos de cumplimiento.
- AWS Audit Manager— Esto le Servicio de AWS ayuda a auditar continuamente su AWS uso para simplificar la gestión del riesgo y el cumplimiento de las normativas y los estándares del sector.

Resiliencia en Amazon Lookout for Vision

La infraestructura AWS global se basa en AWS regiones y zonas de disponibilidad. AWS Las regiones proporcionan varias zonas de disponibilidad aisladas y separadas físicamente, que están conectadas mediante redes de baja latencia, alto rendimiento y alta redundancia. Con las zonas de disponibilidad, puede diseñar y utilizar aplicaciones y bases de datos que realizan una conmutación por error automática entre las zonas sin interrupciones. Las zonas de disponibilidad tienen una mayor disponibilidad, tolerancia a errores y escalabilidad que las infraestructuras tradicionales de uno o varios centros de datos.

Para obtener más información sobre AWS las regiones y las zonas de disponibilidad, consulte Infraestructura global.AWS

Seguridad de la infraestructura en Amazon Lookout for Vision

Como servicio gestionado, Amazon Lookout for Vision está protegido AWS por la seguridad de la red global. Para obtener información sobre los servicios AWS de seguridad y cómo se AWS protege la infraestructura, consulte Seguridad AWS en la nube. Para diseñar su AWS entorno utilizando las mejores prácticas de seguridad de la infraestructura, consulte Protección de infraestructuras en un marco de buena AWS arquitectura basado en el pilar de la seguridad.

Utiliza las llamadas a la API AWS publicadas para acceder a Lookout for Vision a través de la red. Los clientes deben admitir lo siguiente:

- Seguridad de la capa de transporte (TLS). Exigimos TLS 1.2 y recomendamos TLS 1.3.

- Conjuntos de cifrado con confidencialidad directa total (PFS) como DHE (Ephemeral Diffie-Hellman) o ECDHE (Elliptic Curve Ephemeral Diffie-Hellman). La mayoría de los sistemas modernos como Java 7 y posteriores son compatibles con estos modos.

Además, las solicitudes deben estar firmadas mediante un ID de clave de acceso y una clave de acceso secreta que esté asociada a una entidad principal de IAM. También puedes utilizar AWS Security Token Service (AWS STS) para generar credenciales de seguridad temporales para firmar solicitudes.

Monitorización de Amazon Lookout for Vision

La monitorización es una parte importante del mantenimiento de la fiabilidad, la disponibilidad y el rendimiento de Amazon Lookout for Vision y de las demás soluciones de AWS. AWS ofrece las siguientes herramientas de monitorización para monitorizar Lookout for Vision, informar cuando algo no funciona y realizar acciones automáticas cuando proceda:

- Amazon CloudWatch monitorea tus AWS recursos y las aplicaciones en las que AWS ejecutas en tiempo real. Puede recopilar métricas y realizar un seguimiento de las métricas, crear paneles personalizados y definir alarmas que le advierten o que toman medidas cuando una métrica determinada alcanza el umbral que se especifique. Por ejemplo, puedes CloudWatch hacer un seguimiento del uso de la CPU u otras métricas de tus EC2 instancias de Amazon y lanzar automáticamente nuevas instancias cuando sea necesario. Para obtener más información, consulta la Guía del CloudWatch usuario de Amazon.
- Amazon CloudWatch Logs le permite supervisar, almacenar y acceder a sus archivos de registro desde EC2 instancias de Amazon y otras fuentes. CloudTrail CloudWatch Los registros pueden monitorear la información de los archivos de registro y notificarle cuando se alcanzan ciertos umbrales. También se pueden archivar los datos del registro en un almacenamiento de larga duración. Para obtener más información, consulta la Guía del usuario CloudWatch de Amazon Logs.
- Amazon se EventBridge puede utilizar para automatizar sus AWS servicios y responder automáticamente a los eventos del sistema, como los problemas de disponibilidad de las aplicaciones o los cambios de recursos. Los eventos de AWS los servicios se entregan EventBridge prácticamente en tiempo real. Puede crear reglas sencillas para indicar qué eventos le resultan de interés, así como qué acciones automatizadas se van a realizar cuando un evento cumple una de las reglas. Para obtener más información, consulta la Guía EventBridge del usuario de Amazon.
- AWS CloudTrailcaptura las llamadas a la API y los eventos relacionados realizados por su AWS cuenta o en su nombre y entrega los archivos de registro a un bucket de Amazon S3 que especifique. Puede identificar qué usuarios y cuentas llamaron AWS, la dirección IP de origen desde la que se realizaron las llamadas y cuándo se produjeron las llamadas. Para obtener más información, consulte la AWS CloudTrail Guía del usuario de .

Supervisión de Lookout for Vision con Amazon CloudWatch

Puedes monitorizar Lookout for Vision CloudWatch con Lookout for Vision, que recopila datos sin procesar y los procesa para convertirlos en métricas legibles prácticamente en tiempo real. Estas estadísticas se mantienen durante 15 meses, de forma que pueda obtener acceso a información histórica y disponer de una mejor perspectiva sobre el desempeño de su aplicación web o servicio. También puede establecer alarmas que vigilen determinados umbrales y enviar notificaciones o realizar acciones cuando se cumplan dichos umbrales. Para obtener más información, consulta la Guía del CloudWatch usuario de Amazon.

El servicio Lookout for Vision informa de las siguientes métricas en el espacio de nombres `AWS/LookoutVision`.

Métrica	Descripción
`DetectedAnomalyCount`	El número de anomalías detectadas en un proyecto Estadísticas válidas: `Sum,Average` Unidad: recuento
`ProcessedImageCount`	El número total de imágenes procesadas por la detección de anomalías Estadísticas válidas: `Sum,Average` Unidad: recuento
`InvalidImageCount`	El número de imágenes que no eran válidas y no podían devolver un resultado Estadísticas válidas: `Sum,Average` Unidad: recuento
`SuccessfulRequestCount`	El número de llamadas correctas a las llamadas a la API Estadísticas válidas: `Sum,Average`

Métrica	Descripción
	Unidad: recuento
`ErrorCount`	La cantidad de errores de API Estadísticas válidas: `Sum,Average` Unidad: recuento
`ThrottledCount`	El número de errores de API que se debieron a la limitación Estadísticas válidas: `Sum,Average` Unidad: recuento
`Time`	El tiempo en milisegundos que tarda Lookout for Vision en calcular la detección de anomalías Estadísticas válidas: `Data Samples,Average` Unidades: milisegundos para las estadísticas `Average` y recuento para las estadísticas `Data Samples`
`MinInferenceUnits`	El número mínimo de unidades de inferencia especificadas durante la solicitud `StartModel` . Estadísticas válidas: `Average` Unidad: recuento
`MaxInferenceUnits`	El número máximo de unidades de inferencia especificadas durante la solicitud `StartModel` . Estadísticas válidas: `Average` Unidad: recuento

Métrica	Descripción
`DesiredInferenceUnits`	El número de unidades de inferencia a las que Lookout for Vision se escala o reduce verticalmente. Estadísticas válidas: `Average` Unidad: recuento
`InServiceInferenceUnits`	El número de unidades de inferencia que utiliza el modelo. Estadísticas válidas: `Average` Se recomienda utilizar la estadística Promedio para obtener el promedio de 1 minuto del número de instancias que se utilizan. Unidad: recuento

Las siguientes dimensiones se admiten para las métricas Lookout for Vision.

Dimensión	Descripción
`ProjectName`	Puede dividir las métricas por proyecto para ver qué proyectos tienen problemas o necesitan actualizarse.
`ModelVersion`	Puede dividir las métricas por proyecto para ver qué proyectos tienen problemas o necesitan actualizarse.

Registro de llamadas a la API de Lookout for Vision con AWS CloudTrail

Amazon Lookout for Vision está integrado AWS CloudTrail con un servicio que proporciona un registro de las acciones realizadas por un usuario, un rol o AWS un servicio de Lookout for Vision. CloudTrail captura todas las llamadas a la API de Lookout for Vision como eventos. Las llamadas capturadas incluyen las llamadas realizadas desde la consola de Lookout for Vision, así como las

llamadas de código realizadas a las operaciones de API de Lookout for Vision. Si crea una ruta, puede habilitar la entrega continua de CloudTrail eventos a un bucket de Amazon S3, incluidos los eventos de Lookout for Vision. Si no configura una ruta, podrá ver los eventos más recientes en la CloudTrail consola, en el historial de eventos. Con la información recopilada por CloudTrail, puedes determinar la solicitud que se realizó a Lookout for Vision, la dirección IP desde la que se realizó la solicitud, quién la hizo, cuándo se realizó y detalles adicionales.

Para obtener más información CloudTrail, consulta la Guía del AWS CloudTrail usuario.

Busque información sobre Lookout for Vision en CloudTrail

CloudTrail está activado en tu AWS cuenta al crearla. Cuando se produce una actividad en Lookout for Vision, esa actividad se registra en CloudTrail un evento junto con AWS otros eventos de servicio en el historial de eventos. Puedes ver, buscar y descargar los eventos recientes en tu AWS cuenta. Para obtener más información, consulte Visualización de eventos con el historial de CloudTrail eventos.

Para tener un registro continuo de los eventos de tu AWS cuenta, incluidos los eventos de Lookout for Vision, crea una ruta. Un rastro permite CloudTrail entregar archivos de registro a un bucket de Amazon S3. De forma predeterminada, cuando se crea un registro de seguimiento en la consola, el registro de seguimiento se aplica a todas las regiones de AWS. La ruta registra los eventos de todas las regiones de la AWS partición y envía los archivos de registro al bucket de Amazon S3 que especifique. Además, puede configurar otros AWS servicios para analizar más a fondo los datos de eventos recopilados en los CloudTrail registros y actuar en función de ellos. Para más información, consulte los siguientes temas:

- Introducción a la creación de registros de seguimiento
- CloudTrail Integraciones y servicios compatibles
- Configuración de las notificaciones de Amazon SNS para CloudTrail
- Recibir archivos de CloudTrail registro de varias regiones y recibir archivos de CloudTrail registro de varias cuentas

Todas las acciones de Lookout for Vision se registran CloudTrail y se documentan en la documentación de referencia de la API de Lookout for Vision. Por ejemplo, las llamadas a `DetectAnomalies` y `StartModel` las acciones generan entradas en los archivos de CloudTrail registro. `CreateProject`

Si monitorizas las llamadas a la API de Amazon Lookout for Vision, es posible que veas llamadas a las APIs siguientes direcciones.

- visión de lookoutvision: StartTriallDetection
- visión de vigilancia: ListTriallDetection
- visión de vigilancia: DescribeTrialDetection

Amazon Lookout for Vision utiliza estas llamadas especiales para respaldar diversas operaciones relacionadas con las pruebas de detección. Para obtener más información, consulte Verificación del modelo con una tarea de pruebas de detección.

Cada entrada de registro o evento contiene información sobre quién generó la solicitud. La información de identidad del usuario le ayuda a determinar lo siguiente:

- Si la solicitud se realizó con credenciales de AWS Identity and Access Management usuario o root.
- Si la solicitud se realizó con credenciales de seguridad temporales de un rol o fue un usuario federado.
- Si la solicitud la realizó otro AWS servicio.

Para obtener más información, consulte el Elemento userIdentity de CloudTrail .

Descripción de las entradas de los archivos de registro de Lookout for Vision

Un rastro es una configuración que permite la entrega de eventos como archivos de registro a un bucket de Amazon S3 que usted especifique. CloudTrail Los archivos de registro contienen una o más entradas de registro. Un evento representa una solicitud única de cualquier fuente e incluye información sobre la acción solicitada, la fecha y la hora de la acción, los parámetros de la solicitud, etc. CloudTrail Los archivos de registro no son un registro ordenado de las llamadas a la API pública, por lo que no aparecen en ningún orden específico.

En el siguiente ejemplo, se muestra una entrada de CloudTrail registro que demuestra la `CreateDataset` acción.

```
{
  "eventVersion": "1.08",
  "userIdentity": {
```

```
    "type": "AssumedRole",
    "principalId": "AROAYN4CJAYDEXAMPLE:user",
    "arn": "arn:aws:sts::123456789012:assumed-role/Admin/MyUser",
    "accountId": "123456789012",
    "accessKeyId": "ASIAYN4CJAYEXAMPLE",
    "sessionContext": {
      "sessionIssuer": {
        "type": "Role",
        "principalId": "AROAYN4CJAYDCGEXAMPLE",
        "arn": "arn:aws:iam::123456789012:role/Admin",
        "accountId": "123456789012",
        "userName": "Admin"
      },
      "webIdFederationData": {},
      "attributes": {
        "mfaAuthenticated": "false",
        "creationDate": "2020-11-20T13:15:09Z"
      }
    }
  },
  "eventTime": "2020-11-20T13:15:43Z",
  "eventSource": "lookoutvision.amazonaws.com",
  "eventName": "CreateDataset",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "128.0.0.1",
  "userAgent": "aws-cli/3",
  "requestParameters": {
    "projectName": "P1",
    "datasetType": "train",
    "datasetSource": {
      "groundTruthManifest": {
        "s3Object": {
          "bucket": "myuser-bucketname",
          "key": "training.manifest"
        }
      }
    },
    "clientToken": "EXAMPLE-0526-47dd-a5d3-2ca975820a34"
  },
  "responseElements": {
    "status": "CREATE_IN_PROGRESS"
  },
  "requestID": "EXAMPLE-15e1-4bc9-be38-cda2537c75bf",
  "eventID": "EXAMPLE-c5e7-43e0-8449-8d9b87e15acb",
```

```
  "readOnly": false,
  "eventType": "AwsApiCall",
  "managementEvent": true,
  "eventCategory": "Management",
  "recipientAccountId": "123456789012"
}
```

Creación de recursos Amazon Lookout for Vision con AWS CloudFormation

Amazon Lookout for Vision está integrado AWS CloudFormation con un servicio que le ayuda a modelar y configurar AWS 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 Amazon Lookout for Vision.

Cuando la utilices AWS CloudFormation, podrás reutilizar la plantilla para configurar tus proyectos de Lookout for Vision de forma coherente y repetida. Solo tiene que describir los proyectos una vez y, luego, aprovisionar los mismos proyectos una y otra vez en varias cuentas y regiones.

Busca Vision for Vision y plantillas AWS CloudFormation

Para aprovisionar y configurar proyectos para Lookout for Vision y sus servicios relacionados, debe entender las plantillas de AWS CloudFormation. Las plantillas son archivos de texto con formato JSON o YAML. Estas plantillas describen los recursos que deseas aprovisionar en tus AWS CloudFormation pilas. 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 Lookout for Vision, incluidos ejemplos de plantillas JSON y YAML, LookoutVision consulta la referencia de tipos de recursos.

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

Acceder a Amazon Lookout for Vision mediante un punto de conexión de interfaz (AWS PrivateLink)

Puede utilizarla AWS PrivateLink para crear una conexión privada entre su VPC y Amazon Lookout for Vision. Puede acceder a Lookout for Vision como si estuviera en su VPC, sin necesidad de utilizar una pasarela de Internet, un dispositivo NAT, una conexión VPN o una conexión. AWS Direct Connect Las instancias de la VPC no necesitan direcciones IP públicas para acceder a Lookout for Vision.

Esta conexión privada se establece mediante la creación de un punto de conexión de interfaz alimentado por AWS PrivateLink. Creamos una interfaz de red de punto de conexión en cada subred habilitada para el punto de conexión de interfaz. Se trata de interfaces de red administradas por el solicitante que sirven como punto de entrada para el tráfico destinado a Lookout for Vision.

Para obtener más información, consulte Acceso directo AWS PrivateLink en la Servicios de AWSAWS PrivateLink guía.

Consideraciones para los puntos de conexión de VPC de Lookout for Vision

Antes de configurar un punto de conexión de interfaz para Lookout for Vision, consulte Consideraciones en la AWS PrivateLink Guía.

Lookout for Vision admite que se hagan llamadas a todas las acciones de la API a través del punto de conexión de interfaz.

Las políticas de punto de conexión de VPC no son compatibles con Lookout for Vision. De forma predeterminada, el acceso completo a Lookout for Vision se permite a través de la interfaz del punto de conexión. Como alternativa, puede asociar un grupo de seguridad a las interfaces de red del punto de conexión para controlar el tráfico a Lookout for Vision a través del punto de conexión de interfaz.

Creación del punto de conexión de VPC de la interfaz para Lookout for Vision

Puede crear un punto final de interfaz para Lookout for Vision mediante la consola Amazon VPC o AWS Command Line Interface el ().AWS CLI Para obtener más información, consulte Creación de un punto de conexión de interfaz en la Guía de AWS PrivateLink .

Cree un punto de conexión de interfaz para Lookout for Vision utilizando los siguientes nombres de servicio:

```
com.amazonaws.region.lookoutvision
```

Si habilita DNS privado para el punto de conexión de la interfaz, puede realizar solicitudes de API a Lookout for Vision usando su nombre de DNS predeterminado para la región. Por ejemplo, `lookoutvision.us-east-1.amazonaws.com`.

Creación de una política del punto de conexión de VPC para Lookout for Vision

Una política de punto de conexión es un recurso de IAM que puede adjuntar a un punto de conexión de interfaz. La política de punto de conexión predeterminada permite acceso completo a las API de Lookout for Vision a través del punto de conexión de interfaz. Para controlar el acceso permitido a las API de Lookout for Vision desde la VPC, adjunte una política de punto de conexión personalizada al punto de conexión de interfaz.

Una política de punto de conexión especifica la siguiente información:

- Las entidades principales que pueden llevar a cabo acciones (Cuentas de AWS, usuarios de IAM y roles de IAM).
- Las acciones que se pueden realizar.
- El recurso en el que se pueden realizar las acciones.

Para obtener más información, consulte Control del acceso a los servicios con políticas de punto de conexión en la Guía del usuario de AWS PrivateLink .

Ejemplo: Política de punto de conexión de VPC para acciones de Lookout for Vision

El siguiente es un ejemplo de una política de un punto de conexión personalizado para Lookout for Vision. Al adjuntar esta política al punto de conexión de la interfaz, especifica que todos los usuarios que tienen acceso al punto de conexión de la interfaz de la VPC pueden llamar a la operación de `DetectAnomalies` API del modelo Lookout for Vision `myModel` asociado al proyecto `myProject`.

```
{
   "Statement":[
      {
         "Principal":"*",
         "Effect":"Allow",
         "Action":[
            "lookoutvision:DetectAnomalies"
         ],
         "Resource": "arn:aws:lookoutvision:us-west-2:123456789012:model/myProject/
myModel"
      }
   ]
}
```

Amazon Lookout for Vision

Las siguientes tablas describen las cuotas actuales en Amazon Lookout for Vision. Para obtener información acerca de las cuotas que se pueden cambiar, consulte AWS Service Quotas.

Cuotas modelo

Las siguientes cuotas se aplican a las pruebas, el entrenamiento y la funcionalidad de un modelo.

Recurso	Cuota
Formato de archivo compatible	Formatos de imagen PNG y JPEG
Dimensión mínima de imagen del archivo de imagen en un bucket de Amazon S3	64 píxeles x 64 píxeles
Dimensión máxima de imagen del archivo de imagen en un bucket de Amazon S3	4096 píxeles X 4096 píxeles es el máximo. Las dimensiones más pequeñas permiten cargar más rápido.
Diferentes dimensiones de imagen de los archivos de imagen utilizados en un proyecto	Todas las imágenes del conjunto de datos deben tener las mismas dimensiones
Tamaño máximo de archivo de imagen en un bucket de Amazon S3	8 MB
Falta de etiquetas	Las imágenes deben etiquetarse como normales o anómalas antes del entrenamiento. Las imágenes sin etiquetas se ignoran durante el entrenamiento.
Número mínimo de imágenes etiquetadas como normales en el conjunto de datos de entrenamiento	10 para un proyecto con conjuntos de datos de entrenamiento y prueba independientes. 20 para un proyecto con un solo conjunto de datos.

Recurso	Cuota
Número mínimo de imágenes etiquetadas como normales en el conjunto de datos de entrenamiento	0 para un proyecto con conjuntos de datos de entrenamiento y prueba independientes. 10 para un proyecto con un solo conjunto de datos.
Número máximo de imágenes en el conjunto de datos de entrenamiento de clasificación	16,000
Número máximo de imágenes en un conjunto de datos de pruebas de clasificación	4.000
Número mínimo de imágenes etiquetadas como normales en el conjunto de datos de prueba	10
Número mínimo de imágenes etiquetadas como anomalías en el conjunto de datos de prueba	10
Número máximo de imágenes en un conjunto de datos de entrenamiento para la localización de anomalías	8000
Número máximo de imágenes en un conjunto de datos de pruebas para la localización de anomalías	800
Número máximo de imágenes en el conjunto de datos de pruebas de detección	2,000
Tamaño máximo del archivo de manifiesto del conjunto de datos	1 GB
Número máximo de conjuntos de datos de entrenamiento en un modelo	1
Tiempo máximo de entrenamiento	24 horas

Recurso	Cuota
Tiempo máximo de prueba	24 horas
Número máximo de etiquetas de anomalías en un proyecto	100
Número máximo de etiquetas de anomalías en una imagen de máscara	20
Número mínimo de imágenes para una etiqueta de anomalía. Para contar, la imagen debe contener solo un tipo de etiqueta de anomalía.	20 para un único proyecto de conjunto de datos. 10 para cada conjunto de datos de un proyecto con conjuntos de datos de entrenami ento y prueba independientes.

Historial de documentación de Amazon Lookout for Vision

En la siguiente tabla se describen los cambios importantes en cada versión de la Guía para desarrolladores de Amazon Lookout for Vision. Para recibir notificaciones sobre los cambios en esta documentación, puede suscribirse a una fuente RSS.

- Última actualización de la documentación: 20 de febrero de 2023

Cambio	Descripción	Fecha
Notificación del fin del soporte	Aviso de fin de soporte: el 31 de octubre de 2025, AWS dejaremos de ofrecer soporte a Amazon Lookout for Vision. Después del 31 de octubre de 2025, ya no podrás acceder a la consola Lookout for Vision ni a los recursos de Lookout for Vision. Para obtener más información, visita esta entrada de blog.	10 de octubre de 2024
Se ha añadido un ejemplo de función de Lambda	Ejemplo que muestra cómo encontrar anomalías con una AWS Lambda función. Para obtener más informaci ón, consulte Búsqueda de anomalías con una función de AWS Lambda.	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, consulta prácticas recomenda das de seguridad en IAM.	8 de febrero de 2023

Se ha añadido un ejemplo de una exportación de conjunto de datos	Se agregó un ejemplo de Python que muestra cómo usar la operación `ListDatasetEntries` para exportar los conjuntos de datos de un proyecto de Amazon Lookout for Vision. Para obtener más información, consulte Exportación de conjuntos de datos de un proyecto (SDK).	2 de diciembre de 2022
Tema de introducción actualizado	Se ha actualizado la sección de introducción para mostrar la creación de un modelo de segmentación de imágenes con un conjunto de datos de ejemplo. Para obtener más información, consulte Introducción a Amazon Lookout for Vision.	20 de octubre de 2022
Tema sobre solución de problemas añadido	Se ha añadido un tema de solución de problemas de entrenamiento de modelos. Para obtener más información, consulte Resolución de pruebas del entrenamiento de un modelo.	17 de octubre de 2022

Se agregó un tema sobre el uso de los trabajos de Amazon SageMaker AI Ground Truth	En lugar de etiquetar las imágenes usted mismo, puede utilizar los trabajos de Amazon SageMaker AI Ground Truth para etiquetar las imágenes para los modelos de clasificación y segmentación de imágenes. Para obtener más información, consulta Cómo usar un trabajo de Amazon SageMaker AI Ground Truth.	19 de agosto de 2022
Amazon Lookout for Vision ahora ofrece la localización de anomalías.	Puede crear un modelo de segmentación que busque las ubicaciones de una imagen en las que hay diferentes tipos de anomalías (como un rasguño, una abolladura o un desgarro), la etiqueta de la anomalía y el tamaño de la anomalía. Para obtener más información, consulte Ejecutar su modelo formado de Amazon Lookout for Vision.	16 de agosto de 2022
Amazon Lookout for Vision ahora proporciona inferencias de CPU en dispositivos de borde.	Los modelos de Amazon Lookout for Vision ahora se pueden implementar para ejecutar inferencias de forma local en una plataforma informática x86 que ejecute Linux con solo una CPU, sin necesidad de un acelerador de GPU. Para obtener más información, consulte Acelerador de CPU.	16 de agosto de 2022

Amazon Lookout for Vision ahora puede escalar automáticamente las unidades de inferencia.	Para hacer frente a los picos de demanda, Etiquetas personalizadas de Amazon Lookout for Vision ahora puede escalar el número de unidades de inferencia que utiliza el modelo. Para obtener más información, consulte Ejecutar su modelo formado de Amazon Lookout for Vision.	16 de agosto de 2022
Se han añadido ejemplos de Java	La guía para desarrolladores de Amazon Lookout for Vision ahora incluye ejemplos de Java. Para obtener más información, consulte Introducción al AWS SDK.	2 de mayo de 2022
Disponibilidad general de la implementación del modelo en un dispositivo de borde	La implementación de modelos en un dispositivo perimetral gestionado por ya AWS IoT Greengrass Version 2 está disponible de forma general. Para obtener más información, consulte Uso del modelo Amazon Lookout for Vision en un dispositivo de borde.	14 de marzo de 2022
Información actualizada sobre el bucket de la consola	Información actualizada sobre el contenido del bucket de consola y enfoques alternativos para crear el bucket de consola. Para obtener más información, consulte Paso 4: Crear el bucket de la consola.	7 de marzo 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 a partir de un archivo CSV. Para obtener más información, consulte Creación de un archivo de manifiesto a partir de un archivo CSV.	10 de febrero de 2022
Obtenga una vista previa de la implementación del modelo en un dispositivo de borde	Ya AWS IoT Greengrass Version 2 está disponible la versión preliminar de la implementación del modelo en un dispositivo perimetral gestionado por. Para obtener más información, consulte Uso del modelo Amazon Lookout for Vision en un dispositivo de borde.	7 de diciembre de 2021
Se han añadido nuevos ejemplos de Python y Java 2	Se agregaron ejemplos de Python y Java 2 para analizar imágenes con `DetectAnomalies` . Para obtener más información, consulte Detección de anomalías en una imagen.	7 de septiembre de 2021
Se han agregado nuevas políticas AWS administradas.	Amazon Lookout for Vision añade compatibilidad con las políticas AWS gestionadas. Para obtener más información, consulte Políticas administradas de AWS para Amazon Lookout for Vision.	11 de mayo de 2021

Información actualizada de la unidad de inferencia.	Se agregó información que describe las unidades de inferencia y cómo se cobran. Para obtener más información, consulte Ejecutar su modelo formado de Amazon Lookout for Vision.	15 de marzo de 2021
Disponibilidad general de Amazon Lookout for Vision.	Amazon Lookout for Vision ya está disponible con carácter general. Ejemplos de código de Python actualizados para gestionar tareas asíncronas, como el entrenamiento de un modelo.	17 de febrero de 2021
Se agregaron el etiquetado y el AWS CloudFormation soporte.	Ahora puedes etiquetar modelos de Amazon Lookout for Vision y crear proyectos AWS CloudFormation con ellos. Para obtener más información, consulte Etiquetado de modelos y Creación de proyectos de Amazon Lookout for Vision con AWS. CloudFormation	31 de enero de 2021
Nueva característica y guía	Esta es la versión inicial del servicio Amazon Lookout for Vision de la Guía para desarrolladores de Amazon Lookout for Vision.	1 de diciembre de 2020

AWS Glosario

Para obtener la AWS terminología más reciente, consulte el AWS glosario de la Glosario de AWS Referencia.