aws

Guía para desarrolladores

Amazon Rekognition



Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon Rekognition: Guía para desarrolladores

Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Las marcas comerciales y la imagen comercial de Amazon no se pueden utilizar en relación con ningún producto o servicio que no sea de Amazon de ninguna manera que pueda causar confusión entre los clientes y que menosprecie o desacredite a Amazon. Todas las demás marcas registradas que no son propiedad de Amazon son propiedad de sus respectivos propietarios, que pueden o no estar afiliados, conectados o patrocinados por Amazon.

Table of Contents

¿Qué es Amazon Rekognition? 1
Capacidades clave 1
Casos de uso 2
Ventajas 4
Amazon Rekognition y aptitud para la HIPAA 5
¿Es la primera vez que usa Amazon Rekognition? 5
Cómo funciona Rekognition 6
Descripción de los tipos de análisis de Rekognition 7
Etiquetas 9
Etiquetas personalizadas 10
Detección de vitalidad facial 11
Detección y análisis faciales 11
Búsqueda de caras 12
Recorridos de las personas 12
Equipos de protección individual 12
Famosos 12
Detección de texto 13
Contenido inapropiado u ofensivo 13
Personalización 13
Análisis masivo 14
Descripción de las operaciones de imágenes y vídeo de Rekognition 14
Operaciones de Amazon Rekognition Image 14
Operaciones de Amazon Rekognition Video 14
Operaciones basadas en almacenamiento y no almacenamiento 15
Uso del SDK de AWS o HTTP para llamar a operaciones de API de Amazon Rekognition 16
Operaciones de API con almacenamiento y sin almacenamiento 16
Operaciones sin almacenamiento 16
Operaciones de API con almacenamiento 19
Descripción del control de versiones de los modelos 20
Introducción 22
Paso 1: Configurar una cuenta de AWS y crear un usuario 22
Crea una AWS cuenta y un usuario 23
Paso 2: Configura el AWS CLI y AWS SDKs 25
Concesión de acceso programático 27

Trabajando con AWS SDKs ... 32
Paso 3: Cómo empezar a usar la API AWS CLI y el AWS SDK ... 33
Formatear los AWS CLI ejemplos ... 34
Siguiente paso ... 34
Paso 4: Introducción al uso de la consola ... 34
Configurar los permisos de la consola ... 35
Ejercicio 1: Detectar objetos y escenas (consola) ... 39
Ejercicio 2: Analizar rostros (consola) ... 46
Ejercicio 3: Comparar rostros (consola) ... 49
Ejercicio 4: Consultar métricas totales (consola) ... 52
Trabajar con imágenes y vídeos ... 55
Trabajar con imágenes ... 55
Especificaciones de imagen ... 56
Análisis de imágenes almacenadas en un bucket de Amazon S3; ... 58
Uso de un sistema de archivos local ... 75
Visualización de cuadros delimitadores ... 90
Obtención de coordenadas de cuadro delimitador y orientación de imagen ... 103
Cómo trabajar con operaciones de análisis de vídeos almacenados ... 114
Tipos de análisis ... 115
Descripción general de la API de Amazon Rekognition Video ... 115
Cómo llamar a las operaciones de Amazon Rekognition Video ... 118
Configuración de Amazon Rekognition Video ... 125
Análisis de un vídeo almacenado (SDK) ... 129
Análisis de un vídeo (AWS CLI) ... 158
Referencia: notificación de resultados de análisis de vídeo ... 162
Solución de problemas de Amazon Rekognition Video ... 163
Trabajar con eventos de vídeo en streaming ... 165
Descripción general de las operaciones del procesador de transmisión de Amazon Rekognition Video ... 166
Agregar etiquetas a un procesador de transmisión nuevo ... 167
Agregar etiquetas a un procesador de transmisión existente ... 168
Listar etiquetas en un procesador de flujo ... 168
Eliminar etiquetas de un procesador de transmisión ... 169
Gestión de errores ... 169
Componentes de un error ... 170
Mensajes y códigos de error ... 170

Control de errores en la aplicación ... 176
Uso de Amazon Rekognition con FedRAMP ... 177
Prácticas recomendadas para sensores, vídeos e imágenes de entrada ... 181
Latencia de operación de Amazon Rekognition Image ... 182
Recomendaciones para la comparación de rostros en las imágenes de entrada ... 182
Recomendaciones generales sobre la introducción de imágenes para operaciones faciales ... 182
Recomendaciones para buscar rostros en una colección ... 183
Recomendaciones de configuración de la cámara (imagen y vídeo) ... 184
Recomendaciones de configuración de la cámara (vídeo almacenado y en streaming) ... 187
Recomendaciones de configuración de la cámara (vídeo en streaming) ... 187
Recomendaciones para el uso de Face Liveness ... 188
Detección de objetos y conceptos ... 189
Etiquetado de objetos de respuesta ... 191
Cuadro delimitador ... 191
Puntuación de confianza ... 192
Elementos principales ... 192
Categorías ... 192
Alias ... 192
Propiedades de la imagen ... 193
Versión del modelo ... 195
Filtros de inclusión y exclusión ... 195
Clasificación y agregación de los resultados ... 196
Detección de etiquetas en una imagen ... 196
DetectLabels solicitud de operación ... 207
DetectLabels respuesta ... 209
Transformar la DetectLabels respuesta ... 212
Detección de etiquetas en un vídeo ... 217
StartLabelDetection Solicita ... 217
GetLabelDetection Respuesta a la operación ... 219
Transformando la GetLabelDetection respuesta ... 225
Detección de etiquetas en eventos de vídeo en streaming ... 234
Configuración de los recursos de Amazon Rekognition Video y Amazon Kinesis ... 235
Llamada a operaciones de detección de etiquetas para la transmisión de eventos de vídeo ... 239
Detección de etiquetas personalizadas ... 246

Detección y análisis de rostros 247
Información general sobre la detección y la comparación de rostros 248
Directrices sobre los atributos faciales 250
Detección de rostros en una imagen 251
DetectFaces solicitud de operación 263
DetectFaces respuesta de operación 263
Comparación de rostros en imágenes 271
CompareFaces solicitud de operación 283
CompareFaces respuesta de operación 283
Detección de rostros en un vídeo almacenado 286
GetFaceDetection respuesta de operación 296
Búsqueda de rostros en una colección 301
Administración de colecciones de rostros, rostros y usuarios 304
Administración de una colección 305
Administración de rostros en una colección 305
Administrar usuarios en una colección 306
Uso de umbrales de similitud para asociar rostros y buscar coincidencias con estos 306
Uso de umbrales de similitud para asociar rostros 306
Uso de umbrales de similitud para que coincidan con rostros 306
Guía para indexar rostros en escenarios comunes 307
Aplicaciones críticas o de seguridad pública 307
Aplicaciones de uso compartido de fotografías y redes sociales 308
Uso general 308
Búsqueda de rostros y usuarios dentro de una colección 308
Casos de uso para fines de seguridad pública 309
Uso de Amazon Rekognition para mejorar la seguridad pública 311
Creación de una colección 312
CreateCollection solicitud de operación 318
CreateCollection respuesta de operación 318
Etiquetado de colecciones 319
Agregar etiquetas a una colección nueva 319
Agregar etiquetas a una colección existente 320
Enumerar las etiquetas de una colección 321
Eliminar etiquetas de una colección 323
Listado de colecciones 323
ListCollections solicitud de operación 330

ListCollections respuesta de operación 330
Descripción de una colección 331
DescribeCollection solicitud de operación 338
DescribeCollection respuesta de operación 338
Eliminación de una colección 339
DeleteCollection solicitud de operación 345
DeleteCollection respuesta de operación 345
Agregar rostros a una colección 346
Filtrado de rostros 347
IndexFaces solicitud de operación 356
IndexFaces respuesta de operación 357
Enumerar rostros y usuarios asociados en una colección 365
ListFaces solicitud de operación 371
ListFaces respuesta de operación 372
Eliminación de rostros de una colección 373
DeleteFaces solicitud de operación 379
DeleteFaces respuesta de operación 379
Creación de un usuario 380
Eliminar un usuario 384
Asociar caras a un usuario 386
AssociateFaces respuesta de operación 389
Desasociar rostros de un usuario 390
DisassociateFaces respuesta de operación 393
Listado de usuarios en una colección 394
ListUsers respuesta de operación 397
Búsqueda de un rostro (ID de rostro) 398
SearchFaces solicitud de operación 404
SearchFaces respuesta de operación 405
Búsqueda de un rostro (imagen) 406
SearchFacesByImage solicitud de operación 414
SearchFacesByImage respuesta de operación 414
Búsqueda de usuarios (ID de rostro/ID de usuario) 415
SearchUsers solicitud de operación 419
SearchUsers respuesta de operación 420
Búsqueda de usuarios (imagen) 421
SearchUsersByImage solicitud de operación 425

SearchUsersByImage respuesta de operación 425
Búsqueda de rostros en vídeos almacenados 427
GetFaceSearch respuesta de operación 436
Búsqueda de rostros en una colección en streaming de vídeo 441
Configuración de los recursos de Amazon Rekognition Video y Amazon Kinesis 442
Búsqueda de rostros en una transmisión de vídeo 445
Transmisión mediante un GStreamer complemento 469
Solución de problemas de vídeo en streaming 472
Recorridos de las personas 479
Respuesta de la operación GetPersonTracking 489
Detección de equipos de protección individual 493
Tipos de EPI 494
Cubierta facial 494
Cubierta para la mano 494
Cubierta para la cabeza 494
Confianza de detección de EPI 495
Descripción de EPI detectado en una imagen 495
Tutorial: Creación de una AWS Lambda función que detecte imágenes con PPE 496
Comprensión de la API de detección de EPI 496
Suministrar una imagen 496
Entender la DetectProtectiveEquipment respuesta 498
Detección de EPI en una imagen 503
Ejemplo: recuadros delimitadores y mascarillas 515
Reconocimiento de famosos 531
Reconocimiento de famosos comparado con la búsqueda de rostros 532
Reconocimiento de famosos en una imagen 532
¿Llamando RecognizeCelebrities 533
RecognizeCelebrities solicitud de operación 543
RecognizeCelebrities respuesta de operación 543
Reconocimiento de famosos en un vídeo almacenado 546
GetCelebrityRecognition respuesta de operación 562
Obtención de información de un famoso 564
¿Llamando GetCelebrityInfo 564
GetCelebrityInfo solicitud de operación 569
GetCelebrityInfo respuesta de operación 570
Moderación del contenido 571

Uso de la moderación de imágenes y vídeos APIs 573
Categorías de etiquetas 573
Tipo de contenido 585
Confianza 586
Control de versiones 586
Clasificación y agregación 586
Estados de adaptador de Moderación personalizada 587
Cómo probar la versión 7 de Moderación de contenido y transformar la respuesta de la API 587
AWS SDK y guía de uso para la moderación de contenido, versión 7 588
Asignaciones de etiquetas para las versiones 6.1 a 7 589
Detección de imágenes inapropiadas 594
Detección de contenido inapropiado en una imagen 594
DetectModerationLabels solicitud de operación 601
DetectModerationLabels respuesta de operación 602
Detectar vídeos almacenados inapropiados 603
GetContentModeration respuesta de operación 611
Mejora de la precisión con la moderación personalizada 614
Crear y usar adaptadores 615
Preparación de los conjuntos de datos de entrada 619
Administración de adaptadores con la AWS CLI y SDKs 621
Tutorial sobre el adaptador de moderación personalizado 628
Evaluación y mejora del adaptador 646
Formatos de archivo de manifiesto 648
Prácticas recomendadas de adaptadores de entrenamiento 653
Configuración de permisos AutoUpdate 654
AWS Notificación de Health Dashboard para Rekognition 657
Revisión de contenido inapropiado con Amazon A2I 658
Detección de texto 664
Detección de texto en una imagen 666
DetectText solicitud de operación 675
DetectText respuesta de operación 676
Detección de texto en un vídeo almacenado 681
Filtros 691
GetTextDetection respuesta 692
Detección de segmentos de vídeo 698
Tomas técnicas 699

Fotogramas negros ... 699
Créditos ... 699
Barras de color ... 700
Caretas ... 700
Logotipos de estudio ... 700
Contenidos ... 700
Detección de tomas ... 701
Acerca de la API de detección de segmentos de Amazon Rekognition Video ... 702
Uso de la API de segmentos de Amazon Rekognition ... 702
Inicio del análisis de segmentos ... 703
Obtención de los resultados del análisis de segmentos ... 704
Ejemplo: Detección de segmentos en un vídeo almacenado ... 709
Detección de la pruebas de vida del rostro ... 722
Requisitos de Face Liveness por parte del usuario ... 724
Diagramas de arquitectura y secuencia ... 725
Requisitos previos ... 727
Paso 1: Configurar una cuenta AWS ... 727
Paso 2: Configura la viveza facial AWS SDKs ... 727
Paso 3: Configurar AWS Amplify Resources ... 728
Prácticas recomendadas para detectar la vitalidad del rostro ... 728
Programación de Amazon Rekognition Face Liveness APIs ... 728
Paso 1: CreateFaceLivenessSession ... 729
Paso 2: StartFaceLivenessSession ... 730
Paso 3: GetFaceLivenessSessionResults ... 731
Paso 4: Responder a los resultados ... 732
Llamando a la cara vivacidad APIs ... 732
Configuración y personalización de la aplicación ... 738
Configuración de su aplicación ... 738
Personalizar su aplicación ... 739
Modelo de responsabilidad compartida de Face Liveness ... 739
Directrices de actualización de Face Liveness ... 744
Control de versiones y plazos ... 744
Matriz de versiones, lanzamientos y compatibilidad ... 745
Comunicación de las nuevas versiones ... 746
Preguntas frecuentes sobre Face Liveness ... 746
Análisis masivo ... 750

Procesamiento de imágenes de forma masiva 750
Para crear un trabajo de análisis masivo (CLI) 750
Manifiestos de salida de Análisis masivo 752
Tipo de contenido 753
Verificación de predicciones y entrenamiento de adaptadores 753
Tutoriales 754
Almacenamiento de datos de Amazon Rekognition con Amazon RDS y DynamoDB 754
Requisitos previos 755
Cómo obtener etiquetas para imágenes en bucket de Amazon S3 755
Creación de una tabla de Amazon DynamoDB 757
Carga de datos a DynamoDB 758
Creación de una base de datos de MySQL en Amazon RDS 760
Carga de datos a una tabla MySQL de Amazon RDS 761
Uso de Amazon Rekognition y Lambda para etiquetar activos en un bucket de Amazon S3 763
Requisitos previos 765
Configuración del rol de IAM para Lambda 765
Creación del proyecto 766
Escribir el código 769
Empaquetar el proyecto 779
Implementación de la función de Lambda 780
Probar el método Lambda 781
Creación de aplicaciones de análisis de vídeo de AWS 782
Requisitos previos 783
Procedimiento 783
Creación de una función de Lambda de Amazon Rekognition 784
Requisitos previos 786
Creación del tema de SNS 786
Crear la función de Lambda 786
Configuración de la función de Lambda 787
Configuración del rol de IAM para Lambda 788
Crear el proyecto AWS Toolkit for Eclipse Lambda 788
Prueba de la función de Lambda 792
Uso de Amazon Rekognition para la verificación de identidad 794
Requisitos previos 795
Creación de una colección 795
Nuevo registro de usuarios 796

Inicio de sesión de usuario existente 805
Uso de Amazon Rekognition (REK) para detectar etiquetas de aplicaciones de marketing 807
Requisitos previos 809
Cómo crear y rellenar una tabla de DynamoDB 809
Creación de una función de Lambda con registro y notificación 812
Configuración de los permisos y la función de Lambda 814
Carga de imágenes en Amazon S3 con metadatos 816
CloudWatch Registros de sondeos del cliente 817
Detección de etiquetas en una imagen mediante Lambda y Python 818
Creación de una función de Lambda (consola) 819
(Opcional) Crear una capa (consola) 821
Añadir código Python (consola) 822
Cómo añadir código Python (consola) 824
Ejemplos de código 828
Conceptos básicos 829
Introducción a Amazon Rekognition 830
Acciones 833
Escenarios 1006
Compilar una colección y encontrar rostros en ella 1007
Creación de una aplicación sin servidor para administrar fotos 1019
Detección de EPI en imágenes 1023
Detectar y mostrar elementos en las imágenes 1024
Detectar rostros en una imagen 1040
Detectar información en vídeos 1041
Detectar objetos en imágenes 1081
Detección de personas y objetos en un video 1084
Guarde EXIF y otra información de la imagen 1085
referencia de la API 1087
Seguridad 1088
Identity and Access Management 1088
Público 1089
Autenticación con identidades 1089
Administración del acceso con políticas 1090
Cómo funciona Amazon Rekognition con IAM 1092
Políticas administradas de AWS 1098
Prácticas recomendadas sobre las políticas 1107

Uso de la consola ... 1108
Ejemplos de uso de políticas basadas en identidades ... 1108
Ejemplos de políticas basadas en recursos ... 1111
Resolución de problemas ... 1112
Protección de los datos ... 1115
Cifrado de datos ... 1116
Privacidad del tráfico entre redes ... 1119
Uso de Amazon Rekognition con puntos de conexión de Amazon VPC ... 1119
Creación de puntos de conexión de la VPC para Amazon Rekognition ... 1120
Creación de una política de punto de conexión de VPC para Amazon Rekognition ... 1121
Validación de conformidad ... 1122
Resiliencia ... 1123
Configuración y análisis de vulnerabilidades ... 1123
Prevención de la sustitución confusa entre servicios ... 1123
Seguridad de la infraestructura ... 1125
Supervisión ... 1126
Monitorización de Rekognition con Amazon CloudWatch ... 1127
Uso de CloudWatch métricas para Rekognition ... 1127
Acceso a métricas de Rekognition ... 1128
Crear una alarma ... 1129
CloudWatch métricas de Rekognition ... 1131
Registrar llamadas a la API Amazon Rekognition con AWS CloudTrail ... 1136
Información de Amazon Rekognition en CloudTrail ... 1137
La descripción de las entradas de archivos de registro de Amazon Rekognition ... 1138
Directrices y cuotas ... 1141
Regiones admitidas ... 1141
Cuotas establecidas ... 1141
Amazon Rekognition Image ... 1141
Análisis masivo de Amazon Rekognition Image ... 1141
Vídeo almacenado en Amazon Rekognition Video ... 1143
Vídeo de streaming de Amazon Rekognition Video ... 1143
Cuotas predeterminadas ... 1143
Calcular el cambio de cuota de TPS ... 1144
Prácticas recomendadas para las cuotas de TPS ... 1144
Crear un caso para cambiar las cuotas de TPS ... 1145
Historial de revisión ... 1148

AWSGlosario de 1163
..... mclxiv

¿Qué es Amazon Rekognition?

Amazon Rekognition es un servicio de análisis de imágenes y vídeos basado en la nube que facilita la incorporación de capacidades avanzadas de visión artificial a sus aplicaciones. El servicio utiliza tecnología de aprendizaje profundo de eficacia probada y no requiere experiencia en machine learning para usarlo. Amazon Rekognition incluye un API sencillo y fácil de utilizar que puede analizar rápidamente cualquier archivo de imagen o vídeo almacenado en Amazon S3.

Puede agregar características que detectan objetos, texto y contenido no seguro, que analizan imágenes o vídeos y que comparan rostros con su aplicación mediante las API de Rekognition. Con las API de reconocimiento facial de Amazon Rekognition Puede detectar, analizar y comparar rostros para una amplia variedad de casos de uso, como la verificación de usuarios, catalogación, contabilización de personas y seguridad pública.

El servicio se basa en la misma tecnología de aprendizaje profundo de eficacia probada y altamente escalable desarrollada por los científicos de visión artificial de Amazon, que puede analizar miles de millones de imágenes y vídeos al día. Rekognition aprende de forma sistemática de los nuevos datos y, con frecuencia, agregamos nuevas etiquetas y características al servicio.

Para obtener más información, consulte Preguntas frecuentes sobre Amazon Rekognition.

Capacidades clave

Análisis de imágenes:

- Detección de objetos, escenas y conceptos: detecta y clasifica objetos, escenas, conceptos y famosos en imágenes.
- Detección de texto: detecta y reconoce texto impreso y manuscrito en imágenes en varios idiomas.
- Contenido no seguro: detecta y filtra contenido e imágenes explícitos, inapropiados y violentos. Detecta etiquetas de contenido no seguro con amplio detalle.
- Reconocimiento de famosos: reconoce en las imágenes a decenas de miles de famosos de distintas categorías, como políticos, atletas, actores y músicos.
- Análisis facial: detecta, analiza y compara rostros y atributos faciales, como el género, la edad y las emociones. Entre los casos de uso pueden incluirse la verificación de usuarios, la catalogación, el conteo de personas y la seguridad pública.

- Etiquetas personalizadas: cree clasificadores personalizados a fin de detectar objetos específicos para su caso de uso, como logotipos, productos o personajes.
- Propiedades de la imagen: analiza propiedades de la imagen como la calidad, el color, la nitidez y el contraste.

Análisis de vídeo:

- Detección de objetos, escenas y conceptos: detecta y clasifica objetos, escenas, conceptos y famosos en vídeos.
- Detección de texto: detecta y reconoce texto impreso y manuscrito en vídeos en varios idiomas.
- Recorridos de las personas: rastrea a las personas identificadas a medida que se mueven en los fotogramas de vídeo.
- Análisis facial: detecta, analiza y compara rostros en vídeos almacenados o transmisiones de vídeo.
- Reconocimiento de famosos: reconoce en los vídeos almacenados a decenas de miles de famosos de distintas categorías, como políticos, atletas, actores y músicos.
- Detección de contenido no seguro: detecta contenido explícito, inapropiado y violento en los vídeos.
- Segmentación de vídeo: identifica de forma automática segmentos de vídeo útiles, como fotogramas negros y créditos finales.
- Face Liveness: detecta si un usuario está presente durante la verificación facial.

Casos de uso

Bibliotecas multimedia que permiten realizar búsquedas: Rekognition detecta etiquetas, objetos, conceptos y escenas en las imágenes y los vídeos. Puede permitir realizar búsquedas en estas etiquetas en función del análisis de contenido visual. Es útil para crear bibliotecas de imágenes y vídeos con capacidad de búsqueda.

Verificación de la identidad del usuario basada en el rostro: confirme la identidad de los usuarios al comparar los rostros de las imágenes con imágenes de rostros de referencia. Resulta útil para verificar la identidad en las aplicaciones.

Detección de Face Liveness: Rekognition Face Liveness es una característica de machine learning (ML) totalmente administrada que se ha diseñado para ayudar a los desarrolladores a impedir el

fraude en la verificación de identidad basada en el rostro. Esta característica le ayuda a comprobar que un usuario está físicamente presente frente a la cámara y que no es un mal actor que esté falsificando su rostro. El uso de Rekognition Face Liveness puede ayudarle a detectar ataques falsos presentados ante una cámara, como fotos impresas, fotos/vídeos digitales o máscaras 3D. También ayuda a detectar los ataques de suplantación de identidad que eluden la cámara, como los vídeos pregrabados o falsos que se inyectan directamente en el subsistema de captura de vídeo.

Búsqueda facial: con Rekognition puede buscar en imágenes, vídeos almacenados y transmisiones de vídeos para detectar rostros que coincidan con los almacenados en un contenedor conocido como colección de rostros. Una colección de rostros es un índice de rostros que usted posee y administra. Para buscar personas en función de su rostro, debe indexar los rostros y, después, buscarlos.

Detección de contenido no seguro: detecta y filtra contenido explícito, inapropiado y violento en imágenes y vídeos. Utiliza etiquetas para un filtrado más detallado en función de las necesidades empresariales. La API de moderación de contenido también devuelve una lista jerárquica de las etiquetas detectadas (objetos y conceptos), junto con las puntuaciones de confianza. Estos objetos o etiquetas indican categorías específicas de contenido no seguro, lo que permite filtrar con amplio detalle y administrar grandes volúmenes de contenido generado por los usuarios (UGC); El resultado de la API de moderación de contenido se puede personalizar con los adaptadores, que mejoran el rendimiento de imágenes como las que se proporcionan como datos de entrenamiento.

Detección de equipos de protección individual: detecte equipos de protección individual en las imágenes para supervisar el cumplimiento de la seguridad en diversos sectores. Las condiciones no seguras pueden marcarse de forma automática al detectar equipos no adecuados y recibir alertas sobre estas condiciones, lo que permite mejorar el cumplimiento y el entrenamiento.

Reconocimiento de famosos: reconoce en sus imágenes y vídeos a famosos de distintas categorías, como políticos, atletas, actores y músicos. Puede identificar la aparición de famosos sin necesidad de proporcionar nombres.

Detección de texto: detecte y extraiga texto de las imágenes para realizar búsquedas visuales o extraer metadatos. Esta operación funciona en diferentes fuentes y estilos. Detecta la orientación para gestionar el texto de los letreros y los banners.

Etiquetas personalizadas: identifica escenas, conceptos y objetos personalizados que son específicos de casos de uso empresariales, como la detección de logotipos. Puede entrenar clasificadores personalizados para gestionar los objetos especializados o exclusivos, lo que mejora la precisión en los objetos clave frente a los clasificadores generales. Para más información,

consulte What is Amazon Rekognition Custom Labels? en la Guía para desarrolladores de Etiquetas personalizadas de Amazon Rekognition.

Ventajas

Integración de análisis de imágenes y vídeos de gran eficacia en la aplicación: agregue a las aplicaciones un análisis preciso de imágenes y vídeos sin necesidad de experiencia. La API de Amazon Rekognition permite el análisis mediante aprendizaje profundo sin necesidad de tener conocimientos de machine learning. Puede integrar la visión artificial en aplicaciones web, móviles y de dispositivos de forma rápida.

Análisis de imágenes y vídeos basado en aprendizaje profundo: analiza las imágenes y los vídeos mediante el aprendizaje profundo para obtener una gran precisión. Amazon Rekognition puede detectar etiquetas, objetos, escenas, rostros y famosos. Filtre los resultados para incluir o excluir etiquetas específicas.

Análisis de imágenes escalable: analiza millones de imágenes para organizar enormes conjuntos de datos visuales. Se escala para gestionar el tráfico y las bibliotecas de imágenes en aumento. No es necesario planificar la capacidad y solo se paga por lo que se usa.

Análisis y filtrado de imágenes en función de propiedades: analice y filtre las imágenes por propiedades como la calidad, el color y el contenido visual, y detecte la nitidez, el brillo y el contraste de las imágenes.

Integración con otros servicios de AWS: Amazon Rekognition se integra de forma inmediata con S3 y Lambda. Puede llamar a las API de Amazon Rekognition desde Lambda y procesar imágenes en Amazon S3 sin mover los datos. Rekognition incorpora escalabilidad y seguridad integradas mediante AWS IAM.

Bajo costo: precio de pago por uso, sin mínimos ni compromisos. Tiene a su disposición el nivel gratuito para comenzar. Ahorre más a medida que se escala el uso con los precios por niveles. Rentable en comparación con las soluciones internas.

Personalización sencilla: personalice la precisión para su caso de uso con los adaptadores. Proporcione imágenes de muestra para entrenar los adaptadores. Mejora la detección de objetos y etiquetas en dominios determinados. Ofrece una forma sencilla de personalizar el análisis sin tener experiencia en ML.

Para obtener más información, consulte Preguntas frecuentes sobre Amazon Rekognition.

Amazon Rekognition y aptitud para la HIPAA

Se trata de un servicio compatible con HIPAA. Para obtener más información acerca de AWS, la ley de responsabilidad y portabilidad de seguros médicos de Estados Unidos de 1996 (HIPAA) y el uso de los servicios de AWS para procesar, almacenar y transmitir información sanitaria protegida (PHI), consulte Información general sobre la HIPAA.

¿Es la primera vez que usa Amazon Rekognition?

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

1. Cómo funciona Amazon Rekognition – esta sección presenta varios componentes de Amazon Rekognition que se utilizan para crear una experiencia integral.
2. Introducción a Amazon Rekognition – en esta sección, configura su cuenta, instala el SDK que refleja el idioma que elija y prueba la API Amazon Rekognition. Para obtener una lista de los lenguajes de programación admitidos por Amazon Rekognition, consulte Uso de Rekognition con un SDK AWS.
3. Trabajar con imágenes – esta sección proporciona información sobre el uso de Amazon Rekognition con imágenes almacenadas en buckets de Amazon S3 e imágenes cargadas a partir de un sistema de archivos local.
4. Cómo trabajar con operaciones de análisis de vídeos almacenados – esta sección proporciona información sobre el uso de Amazon Rekognition con vídeos almacenados en un bucket de Amazon S3.
5. Trabajar con eventos de vídeo en streaming – esta sección proporciona información sobre el uso de Amazon Rekognition con vídeos en streaming.

Cómo funciona Amazon Rekognition

Amazon Rekognition proporciona dos conjuntos de API para el análisis visual:

- Amazon Rekognition Image para el análisis de imágenes
- Amazon Rekognition Video para el análisis de vídeo

Análisis de imágenes

Amazon Rekognition Image permite a sus aplicaciones:

- Detectar objetos, escenas y conceptos en las imágenes
- Reconocer famosos
- Detectar texto en distintos idiomas
- Detectar contenido o imágenes explícitos, inapropiados o violentos
- Detectar, analizar y comparar rostros y atributos faciales como la edad y las emociones
- Detectar la presencia de equipos de protección individual

Entre los casos de uso se incluyen la mejora de las aplicaciones de fotografía, la catalogación de imágenes y la moderación de contenido.

Análisis de vídeo

Amazon Rekognition Video permite a sus aplicaciones:

- Realizar un seguimiento de personas y objetos en distintos fotogramas de vídeo
- Reconocer objetos
- Reconocer famosos
- Buscar personas de interés en vídeos almacenados y transmisiones de vídeo
- Analizar atributos como la edad y las emociones en los rostros
- Detectar contenido o imágenes explícitos, inapropiados o violentos
- Agregar y clasificar los resultados de los análisis por marcas de tiempo y segmentos
- Detectar personas, mascotas y paquetes en la transmisión de vídeo

Entre los casos de uso se incluyen el análisis de vídeo, la catalogación de vídeos y el filtrado de contenido inapropiado.

Características principales

- Análisis de aprendizaje profundo de gran eficacia
- Detección de alta precisión de objetos, escenas, rostros y texto
- API fácil de usar para la integración en aplicaciones
- Modelos personalizables ajustados a sus datos
- Análisis escalable de bibliotecas multimedia

Amazon Rekognition permite mejorar la precisión de algunos modelos de aprendizaje profundo mediante el entrenamiento de un adaptador personalizado. Por ejemplo, con Moderación personalizada de Amazon Rekognition puede adaptar el modelo de análisis de imágenes base de Amazon Rekognition mediante el entrenamiento de un adaptador personalizado con sus imágenes. Consulte Mejora de la precisión con la moderación personalizada para obtener más información.

Las siguientes secciones abordan los tipos de análisis que proporciona Amazon Rekognition, así como información general de las operaciones de Amazon Rekognition Image y Amazon Rekognition Video. También se trata la diferencia entre operaciones sin almacenamiento y con almacenamiento.

Para ver una demostración de Amazon Rekognition APIs, consulte el paso 3: Cómo empezar a utilizar la AWS CLI y la API del SDK de AWS, que incluye la prueba de Rekognition en la consola. AWS

Temas

- Descripción de los tipos de análisis de Rekognition
- Descripción de las operaciones de imágenes y vídeo de Rekognition
- Descripción de operaciones de API con almacenamiento y sin almacenamiento
- Descripción del control de versiones de los modelos

Descripción de los tipos de análisis de Rekognition

Los siguientes son los tipos de análisis que pueden realizar la API de Amazon Rekognition Image y la API de Amazon Rekognition Video. Para obtener información acerca de, consulte. APIs Descripción de las operaciones de imágenes y vídeo de Rekognition

En la siguiente tabla se enumeran las operaciones que debe utilizar en relación con el tipo de medio con el que trabaja y su caso de uso:

Caso de uso	Tipo de medios	Operaciones
Moderación del contenido	Imágenes	DetectModerationLabels, StartMediaAnalysisJob, GetMediaAnalysisJob, ListMediaAnalysisJobs
	Vídeo almacenado	StartContentModeration, GetContentModeration
Verificación de identidad	Imágenes	CreateCollection, CreateUser, IndexFaces, AssociateFaces, SearchFacesByImage, SearchUsersByImage
	Vídeo almacenado	CreateCollection, IndexFaces, StartFaceSearch, GetFaceSearch
	Streaming de vídeo (Detección de la pruebas de vida del rostro)	CreateFaceLivenessSession, StartFaceLivenessSession, GetFaceLivenessSessionResults,
Análisis facial	Imágenes	DetectFaces, CompareFaces
	Vídeo almacenado	StartFaceDetection, GetFaceDetection
	Streaming de vídeo	CreateStreamProcessor, StartStreamProcessor
Reconocimiento de objetos y actividades	Imágenes	DetectLabels

Caso de uso	Tipo de medios	Operaciones
	Vídeo almacenado	StartLabelDetection, GetLabelDetection
Hogar conectado	Streaming de vídeo	StartStreamProcessor
Análisis de medios	Vídeo almacenado	StartSegmentDetection, GetSegmentDetection
Seguridad laboral	Imágenes	DetectProtectiveEquipment
Detección de texto	Imágenes	DetectText
	Video	StartTextDetection, GetTextDetection
Recorridos de las personas	Video	StartPersonTracking, GetPersonTracking
Reconocimiento de famosos	Imágenes	RecognizeCelebrities
	Video	StartCelebrityRecognition, GetCelebrityRecognition
Detección de etiquetas personalizadas	Imágenes	DetectCustomLabels
	Entrenamiento de modelos	Consulte la guía para desarrolladores de Etiquetas personalizadas

Etiquetas

Una etiqueta hace referencia a cualquiera de los siguientes elementos: objetos (por ejemplo, flor, árbol o mesa), eventos (por ejemplo, una boda, graduación o cumpleaños) o conceptos (por ejemplo, un paisaje, un atardecer y la naturaleza) o actividades (por ejemplo, corriendo o jugando al baloncesto). Amazon Rekognition puede detectar etiquetas en imágenes y vídeos. Para obtener más información, consulte Detección de objetos y conceptos.

Rekognition puede detectar una gran lista de etiquetas en imágenes y vídeo almacenado. Rekognition también puede detectar una pequeña cantidad de etiquetas en la transmisión de vídeo.

Utilice las siguientes operaciones para detectar etiquetas en función de su caso de uso:

- Para detectar etiquetas en las imágenes: utilice DetectLabels. Puede identificar las propiedades de la imagen, como los colores dominantes y la calidad de la imagen. Para ello, utilice DetectLabelswith `IMAGE_PROPERTIES` como parámetro de entrada.
- Para detectar etiquetas en los vídeos almacenados: utilice StartLabelDetection. El vídeo almacenado no admite la detección de los colores y la calidad de imagen dominantes.
- Para detectar etiquetas en la transmisión de vídeo: utilice CreateStreamProcessor. La detección de los colores y la calidad de imagen dominantes no se admite en la transmisión de vídeo.

Puede especificar qué tipos de etiquetas desea que se devuelvan tanto para la detección de etiquetas de imagen como de vídeo almacenado mediante opciones de filtrado inclusivas y exclusivas.

Etiquetas personalizadas

Etiquetas personalizadas de Amazon Rekognition permite identificar los objetos y las escenas que requiere su empresa entrenando a un modelo de machine learning. Por ejemplo, puede entrenar a un modelo para que detecte logotipos o piezas de máquinas de ingeniería en una línea de ensamblaje.

> **Note**
>
> Para obtener más información acerca de Etiquetas personalizadas de Amazon Rekognition, consulte la Guía para desarrolladores de Etiquetas personalizadas de Amazon Rekognition.

Amazon Rekognition proporciona una consola que se utiliza para crear, entrenar, evaluar y ejecutar un modelo de machine learning. Para obtener más información, consulte Introducción a Etiquetas personalizadas de Amazon Rekognition en la Guía para desarrolladores de Etiquetas personalizadas de Amazon Rekognition. También puede utilizar la API de Etiquetas personalizadas de Amazon Rekognition para entrenar a un modelo y ejecutarlo. Para obtener más información, consulte Introducción al SDK de etiquetas personalizadas de Amazon Rekognition en la Guía para desarrolladores de Amazon Rekognition. CustomLabels

Para analizar imágenes mediante un modelo entrenado, utilice. DetectCustomLabels

Detección de vitalidad facial

Amazon Rekognition Face Liveness puede ayudarle a comprobar que un usuario que se está sometiendo a una verificación de identidad basada en el rostro está físicamente presente frente a la cámara y no es un actor malintencionado que intenta suplantar el rostro del usuario. Detecta los ataques simulados que se presentan ante una cámara y los ataques que eluden la cámara. Un usuario puede comprobar la vitalidad de su rostro haciéndose un autorretrato en vídeo y, se obtiene una puntuación de vitalidad para la comprobación. La vitalidad del rostro se determina mediante un cálculo probabilístico y, a continuación, se obtiene una puntuación de confianza (entre 0 y 100) tras la comprobación. Cuanto más alta sea la puntuación, mayor será la confianza en que la persona que recibe el cheque está viva.

Para obtener más información sobre la vitalidad del rostro, consulte Detección de la pruebas de vida del rostro.

Detección y análisis faciales

Amazon Rekognition puede detectar rostros en imágenes y vídeos almacenados. Con Amazon Rekognition, puede obtener información sobre:

- Dónde se detectan rostros en una imagen o un vídeo
- Referencias faciales como, por ejemplo, la posición de los ojos
- La presencia de oclusión facial en las imágenes
- Emociones detectadas, como felicidad o tristeza
- La dirección de la mirada de una persona en las imágenes

También puede interpretar información demográfica como el sexo o la edad. Puede comparar un rostro en una imagen con los rostros detectados en otra imagen. La información sobre rostros también se puede almacenar para recuperación posterior. Para obtener más información, consulte Detección y análisis de rostros.

Para detectar rostros en imágenes, utilice DetectFaces. Para detectar rostros en vídeos almacenados, utilice StartFaceDetection.

Búsqueda de caras

Amazon Rekognition puede buscar rostros. La información facial se indexa en un contenedor conocido como colección. La información de rostros en la colección puede corresponderse con rostros detectados almacenados en imágenes, vídeos almacenados y vídeos en streaming. Para obtener más información, Búsqueda de rostros en una colección.

Para buscar rostros conocidos en imágenes, utilice DetectFaces. Para buscar rostros conocidos en vídeos almacenados, utilice StartFaceDetection. Para buscar rostros conocidos en vídeos en streaming, utilice CreateStreamProcessor.

Recorridos de las personas

Amazon Rekognition puede rastrear las rutas de las personas detectadas en un vídeo almacenado. Amazon Rekognition Video proporciona seguimiento, detalles de rostros e información de ubicación en fotograma de las personas detectadas en un vídeo. Para obtener más información, consulte Recorridos de las personas.

Para detectar personas en vídeos almacenados, utilice StartPersonTracking.

Equipos de protección individual

Amazon Rekognition puede detectar el equipo de protección individual (EPI) que llevan las personas en una imagen. Amazon Rekognition detecta cubiertas faciales, cubiertas para manos y cubiertas para la cabeza. Amazon Rekognition predice si un artículo de EPI cubre la parte del cuerpo adecuada. También puede encontrar casillas delimitadoras para las personas detectadas y los elementos de EPI. Para obtener más información, consulte Detección de equipos de protección individual.

Para detectar el PPE en las imágenes, utilice DetectProtectiveEquipment.

Famosos

Amazon Rekognition puede reconocer miles de famosos en imágenes y en vídeos almacenados. Puede obtener información sobre dónde se encuentra el rostro de un famoso en una imagen, referencias faciales y la postura del rostro de un famoso. Puede obtener información de seguimiento de famosos a medida que aparecen en un vídeo almacenado. También puede obtener más información sobre un famoso reconocido, como la emoción que expresa y la forma en que presenta su género. Para obtener más información, consulte Reconocimiento de famosos.

Para reconocer famosos en imágenes, utilice RecognizeCelebrities. Para reconocer famosos en vídeos almacenados, utilice StartCelebrityRecognition.

Detección de texto

Amazon Rekognition Text in Image puede detectar texto en imágenes y convertirlo en texto legible por una máquina. Para obtener más información, consulte Detección de texto.

Para detectar texto en imágenes, utilice DetectText.

Contenido inapropiado u ofensivo

Amazon Rekognition puede analizar imágenes y vídeos almacenados para detectar contenido violento y para adultos. Para obtener más información, consulte Moderación del contenido.

Para detectar imágenes no seguras, utilice DetectModerationLabels. Para detectar vídeos almacenados no seguros, utilice StartContentModeration.

Personalización

Algunos análisis de imágenes APIs que ofrece Rekognition le permiten mejorar la precisión de los modelos de aprendizaje profundo mediante la creación de adaptadores personalizados entrenados en sus propios datos. Los adaptadores son componentes que se conectan al modelo de aprendizaje profundo previamente entrenado de Rekognition, lo que mejora su precisión con el conocimiento del dominio basado en sus imágenes. Se entrena un adaptador para que se adapte a sus necesidades proporcionando y anotando imágenes de muestra.

Después de crear un adaptador, se le proporciona un. AdapterId Puedes proporcionarlo AdapterId a una operación para especificar que quieres usar el adaptador que has creado. Por ejemplo, se proporciona AdapterId a la DetectModerationLabelsAPI para el análisis sincrónico de imágenes. Si lo proporciona AdapterId como parte de la solicitud, Rekognition lo utilizará automáticamente para mejorar las predicciones de sus imágenes. Esto le permite aprovechar las capacidades de Rekognition y, al mismo tiempo, personalizarlo para que se adapte a sus necesidades.

También tiene la opción de obtener predicciones de imágenes de forma masiva con la API. StartMediaAnalysisJob Para obtener más información, consulte Análisis masivo.

Puede evaluar la precisión de las operaciones de Rekognition cargando imágenes en la consola de Rekognition y realizando un análisis de estas imágenes. Rekognition anotará sus imágenes con la característica seleccionada y, a continuación, podrá revisar las predicciones con las predicciones verificadas para determinar qué etiquetas se beneficiarían de la creación de un adaptador.

Actualmente, puede utilizar adaptadores con DetectModerationLabels. Para obtener más información acerca de cómo utilizar el adaptadores, consulte Mejora de la precisión con la moderación personalizada.

Análisis masivo

Rekognition Bulk Analysis le permite procesar una gran colección de imágenes de forma asíncrona mediante un archivo de manifiesto junto con la operación. StartMediaAnalysisJob Para obtener más información, consulte Análisis masivo.

Descripción de las operaciones de imágenes y vídeo de Rekognition

Amazon Rekognition ofrece dos conjuntos de API principales para el análisis de imágenes y vídeos:

- Amazon Rekognition Image: esta API está diseñada para analizar imágenes.
- Amazon Rekognition Video: esta API se centra en analizar tanto los vídeos almacenados como las transmisiones de vídeo.

Ambos APIs pueden detectar diversas entidades, como rostros y objetos. Para comprender en su totalidad los tipos de comparación y detección compatibles, consulte la sección sobre Descripción de los tipos de análisis de Rekognition.

Operaciones de Amazon Rekognition Image

Las operaciones de Amazon Rekognition Image son sincrónicas. La entrada y la respuesta se encuentran en formato JSON. Las operaciones de Amazon Rekognition Image analizan una imagen de entrada que está en formato de imagen .jpg o .png. La imagen se transfiere a una operación de Amazon Rekognition Image que puede almacenarse en un bucket de Amazon S3. Si no va a utilizar AWS CLI, también puede transferir bytes de imágenes codificados en Base64 directamente a una operación de Amazon Rekognition. Para obtener más información, consulte Cómo trabajar con imágenes.

Operaciones de Amazon Rekognition Video

La API de Amazon Rekognition Video facilita el análisis de los vídeos almacenados en un bucket de Amazon S3 o transmitidos a través de Amazon Kinesis Video Streams.

Para las operaciones de vídeos almacenados, tenga en cuenta lo siguiente:

- Las operaciones son asíncronas.
- El análisis debe iniciarse con una operación de «Inicio» (por ejemplo, StartFaceDetectionpara la detección de rostros en los vídeos almacenados).
- El estado de finalización del análisis se publica en un tema de Amazon SNS.
- Para recuperar los resultados de un análisis, utilice la operación «Obtener» correspondiente (por ejemplo, GetFaceDetection).
- Para obtener más información, consulte Cómo trabajar con el análisis de vídeo almacenado.

Para el análisis de transmisión de vídeo:

- Entre las capacidades se incluyen la búsqueda de rostros en las colecciones de Rekognition Video y la detección de etiquetas (de objeto o concepto).
- Los resultados del análisis de las etiquetas se envían como notificaciones de Amazon SNS y Amazon S3.
- Los resultados de la búsqueda de rostros se envían a un flujo de datos de Kinesis.
- La gestión del análisis de vídeo en streaming se realiza mediante un procesador de streaming Amazon Rekognition Video (por ejemplo, se crea un procesador con). CreateStreamProcessor
- Para obtener más información, consulte Cómo trabajar con transmisión de eventos de vídeo.

Cada operación de análisis de vídeo devuelve metadatos sobre el vídeo que se está analizando, así como un identificador de trabajo y una etiqueta de trabajo. Operaciones como Detección de etiquetas y Moderación de contenido para vídeo permiten ordenar por marca de tiempo o nombre de etiqueta, además de agregar los resultados por marca de tiempo o por segmento.

Operaciones basadas en almacenamiento y no almacenamiento

Las operaciones de Amazon Rekognition se agrupan en las siguientes categorías.

- Operaciones API sin almacenamiento: en estas operaciones, Amazon Rekognition no conserva ninguna información. Usted proporciona las imágenes y vídeos de entrada, la operación realiza el análisis y devuelve los resultados, pero no se guarda nada en Amazon Rekognition. Para obtener más información, consulte Operaciones sin almacenamiento.
- Operaciones API basadas en almacenamiento: los servidores de Amazon Rekognition pueden almacenar información facial detectada en contenedores conocidos como colecciones.

Amazon Rekognition ofrece operaciones API adicionales que puede utilizar para buscar rostros coincidentes en la información de rostros almacenados. Para obtener más información, consulte Operaciones de API con almacenamiento.

Uso del SDK de AWS o HTTP para llamar a operaciones de API de Amazon Rekognition

Puede llamar a las operaciones de API de Amazon Rekognition utilizando el AWS SDK o directamente a través de HTTP. A menos que tenga una buena razón para no hacerlo, debería usar siempre el AWS SDK. Los ejemplos de Java de esta sección usan el SDK de AWS. El archivo de proyecto de Java no se proporciona, pero puede utilizar AWS Toolkit for Eclipse para desarrollar aplicaciones de AWS mediante Java.

En los ejemplos de .NET de esta sección, se utiliza AWS SDK para .NET. Puede utilizar AWS Toolkit for Visual Studio para desarrollar aplicaciones de AWS con .NET. Incluye útiles plantillas, así como AWS Explorer, para la implementación de aplicaciones y la administración de servicios.

En la Reference de la API de esta guía se explica cómo llamar a las operaciones de Amazon Rekognition mediante HTTP. Para información de referencia de Java, consulte AWS SDK para Java.

Los puntos de conexión del servicio Amazon Rekognition que puede utilizar se documentan en AWS Regions and Endpoints.

Cuando llame a Amazon Rekognition con HTTP, utilice operaciones POST HTTP.

Descripción de operaciones de API con almacenamiento y sin almacenamiento

Amazon Rekognition proporciona dos tipos de operaciones de API. Operaciones sin almacenamiento, en las que Amazon Rekognition no almacena ninguna información, y operaciones de almacenamiento, en las que Amazon Rekognition almacena determinada información facial.

Operaciones sin almacenamiento

Amazon Rekognition proporciona las siguientes operaciones de API sin almacenamiento para imágenes:

- DetectLabels

- DetectFaces
- CompareFaces
- DetectModerationLabels
- DetectProtectiveEquipment
- RecognizeCelebrities
- DetectText
- GetCelebrityInfo

Amazon Rekognition proporciona las siguientes operaciones de API sin almacenamiento para vídeos:

- StartLabelDetection
- StartFaceDetection
- StartPersonTracking
- StartCelebrityRecognition
- StartContentModeration

Estas operaciones se denominan operaciones de API sin almacenamiento porque cuando se llama a la operación, Amazon Rekognition no conserva la información detectada sobre la imagen de entrada. Al igual que con el resto de las operaciones de API de Amazon Rekognition las operaciones de API sin almacenamiento no conservan los bytes de las imágenes de entrada.

En el siguiente ejemplo se muestran escenarios en los que podría integrar operaciones de API sin almacenamiento en su aplicación. En estos supuestos se presupone que dispone de un repositorio local de imágenes.

Example 1: Una aplicación que busca imágenes en su repositorio local que contienen etiquetas específicas

En primer lugar, detecta las etiquetas (objetos y conceptos) con la operación `DetectLabels` de Amazon Rekognition en cada una de las imágenes del repositorio y crea un índice en el cliente, como se muestra a continuación:

```
Label        ImageID
```

```
tree          image-1
flower        image-1
mountain      image-1
tulip         image-2
flower        image-2
apple         image-3
```

A continuación, la aplicación puede buscar este índice para encontrar imágenes en el repositorio local que contengan una etiqueta específica. Por ejemplo, las imágenes que contienen un árbol.

Cada etiqueta que detecta Amazon Rekognition tiene un valor de confianza asociado. Este valor indica el grado de confianza de que la imagen de entrada contenga esa etiqueta. Puede utilizar este valor de confianza para realizar un filtrado adicional de las etiquetas en el cliente en función de los requisitos de su aplicación sobre el nivel de confianza de la detección. Por ejemplo, si necesita etiquetas precisas, podría filtrar y seleccionar solo las etiquetas con mayor confianza (por ejemplo, con un 95% o más). Si la aplicación no requiere un valor de confianza alto, podría elegir filtrar las etiquetas con un valor de confianza inferior (cercano al 50%).

Example 2: Una aplicación para mostrar imágenes de rostros mejoradas

En primer lugar, puede detectar rostros en cada una de las imágenes del repositorio local utilizando la operación `DetectFaces` de Amazon Rekognition en el cliente y crear un índice en el cliente. Para cada rostro, la operación devuelve metadatos que incluyen un cuadro delimitador, referencias faciales (por ejemplo, la posición de la boca y la oreja) y atributos faciales (por ejemplo, el sexo). Puede almacenar estos metadatos en un índice local en el cliente, como se muestra a continuación:

```
ImageID     FaceID     FaceMetaData

image-1     face-1     <boundingbox>, etc.
image-1     face-2     <boundingbox>, etc.
image-1     face-3     <boundingbox>, etc.
...
```

En este índice, la clave principal es una combinación de `ImageID` y `FaceID`.

A continuación, puede utilizar la información del índice para mejorar las imágenes cuando la aplicación las muestre desde su repositorio local. Por ejemplo, podría añadir un cuadro delimitador alrededor del rostro o resaltar rasgos faciales.

Operaciones de API con almacenamiento

Amazon Rekognition Image IndexFacesadmite la operación, que puede utilizar para detectar rostros en una imagen y conservar la información sobre los rasgos faciales detectados en una colección de Amazon Rekognition. Este es un ejemplo de una operación de API con almacenamiento porque el servicio conserva la información en el servidor.

Amazon Rekognition Image proporciona las siguientes operaciones de API de almacenamiento:

- IndexFaces
- ListFaces
- SearchFacesByImage
- SearchFaces
- DeleteFaces
- DescribeCollection
- DeleteCollection
- ListCollections
- CreateCollection

Amazon Rekognition Video proporciona las siguientes operaciones de API de almacenamiento:

- StartFaceSearch
- CreateStreamProcessor

Para poder almacenar información facial, primero debe crear una colección de rostros en una de las regiones de AWS de la cuenta. Esta colección de rostros se especifica cuando se llama a la operación `IndexFaces`. Después de crear una colección de rostros y almacenar información de los rasgos faciales de todos los rostros, puede buscar en la colección rostros coincidentes. Por ejemplo, puede detectar el rostro de mayor tamaño en una imagen y buscar rostros coincidentes en una colección llamando a `searchFacesByImage.`

Se puede acceder a la información facial almacenada en colecciones mediante `IndexFaces` para las operaciones de Amazon Rekognition Video. Por ejemplo, en un vídeo puede buscar personas cuyos rostros coincidan con los de una colección existente llamando a StartFaceSearch.

Para obtener más información acerca de cómo se crean y se administran colecciones, consulte Búsqueda de rostros en una colección.

> **Note**
>
> Las colecciones almacenan vectores faciales, que son representaciones matemáticas de los rostros. Las colecciones no almacenan imágenes de rostros.

Example 1: Una aplicación que autentica el acceso a un edificio

Se comienza creando una colección de rostros para almacenar imágenes de credenciales escaneadas mediante la operación `IndexFaces`, que extrae los rostros y los almacena como vectores de imagen en los que es posible realizar búsquedas. A continuación, cuando un empleado entra en el edificio, se captura una imagen del rostro del empleado y se envía a la operación `SearchFacesByImage`. Si el rostro coincidente produce una puntuación de similitud lo suficientemente alta (por ejemplo, un 99%), se puede autenticar al empleado.

Descripción del control de versiones de los modelos

Amazon Rekognition utiliza modelos de aprendizaje profundo para llevar a cabo la detección de rostros y para buscar rostros en colecciones. Sigue mejorando la precisión de sus modelos de acuerdo a los comentarios de los clientes y los avances en la investigación de aprendizaje profundo. Estas mejoras se envían como actualizaciones de modelo. Por ejemplo, con la versión 1.0 del modelo, IndexFaces puede indexar los 15 rostros de mayor tamaño de una imagen. Las versiones posteriores del modelo permiten a `IndexFaces` indexar los 100 rostros de mayor tamaño de una imagen.

Al crear una nueva colección, está asociada a la versión más reciente del modelo. Para mejorar la precisión, el modelo se actualiza ocasionalmente.

Cuando se publica una nueva versión del modelo, sucede lo siguiente:

- Las nuevas colecciones que cree se asocian con el último modelo. Los rostros que añada a las nuevas colecciones utilizando IndexFaces se detectan utilizando el modelo más reciente.
- Las colecciones existentes siguen utilizando la versión del modelo con el que se han creado. Los vectores de rostros almacenados en estas colecciones no se actualizan automáticamente a la última versión del modelo.

- Los nuevos rostros que se añaden a una colección existente se detectan utilizando el modelo que ya está asociado a la colección.

Las distintas versiones del modelo no son compatibles entre sí. En concreto, si una imagen se indexa en varias colecciones que utilizan distintas versiones del modelo, los identificadores de rostro para los mismos rostros detectados son distintos. Si se indexa una imagen en varias colecciones que están asociadas al mismo modelo, los identificadores del rostro son los mismos.

La aplicación podría tener problemas de compatibilidad si la administración de la colección no tiene en cuenta las actualizaciones del modelo. Puede determinar la versión del modelo que utiliza una colección usando el campo `FaceModelVersion` que se devuelve en la respuesta de la operación de colección (por ejemplo, `CreateCollection`). Puede obtener el modelo versión de una colección existente llamando a DescribeCollection. Para obtener más información, consulte Descripción de una colección.

Los vectores de rostro existentes en una colección no se pueden actualizar a una versión posterior del modelo. Dado que Amazon Rekognition no almacena bytes de imagen de origen, puede volver a indexar imágenes automáticamente utilizando una versión posterior del modelo.

Para utilizar el último modelo en rostros que se almacenan en una colección existente, cree una nueva colección (CreateCollection) y vuelva a indexar las imágenes de origen en la nueva colección (`Indexfaces`). Tiene que actualizar los identificadores de rostro almacenados por su aplicación ya que los identificadores de rostro de la nueva colección son distintos de los identificadores de rostro de la colección antigua. Si ya no necesita la colección antigua, puede eliminarla utilizando DeleteCollection.

Las operaciones sin estado, como DetectFaces, usan la última versión del modelo.

Introducción a Amazon Rekognition

En esta sección se proporcionan temas para empezar a utilizar Amazon Rekognition. Si es la primera vez que utiliza Amazon Rekognition le recomendamos que consulte primero los conceptos y la terminología presentados en Cómo funciona Amazon Rekognition.

Antes de poder usar Rekognition, debe crear una cuenta de AWS y obtener un ID de cuenta de AWS. También querrá crear un usuario, que permita al sistema Amazon Rekognition determinar si tiene los permisos necesarios para acceder a sus recursos.

Después de crear sus cuentas, querrá instalar y configurar el AWS CLI y AWS SDKs. AWS CLI Le permite interactuar con Amazon Rekognition y otros servicios a través de la línea de comandos, AWS SDKs mientras que le permite usar lenguajes de programación como Java y Python para interactuar con Amazon Rekognition.

Una vez que haya configurado AWS CLI y AWS SDKs, podrá ver algunos ejemplos de cómo utilizar ambos. También puede ver algunos ejemplos de cómo interactuar con Amazon Rekognition mediante la consola.

La consola de Amazon Rekognition permite utilizar los flujos de trabajo de Análisis masivo y Moderación personalizada. Además de estas capacidades, también puede usar la consola para ver una demostración de las diversas características de Amazon Rekognition. Tenga en cuenta que la API es necesaria para utilizar la mayoría de las características de Amazon Rekognition.

Temas

- Paso 1: Configurar una cuenta de AWS y crear un usuario
- Paso 2: Configure y AWS CLI AWS SDKs
- Paso 3: Cómo empezar a usar la API AWS CLI y el AWS SDK
- Paso 4: Introducción al uso de la consola de Amazon Rekognition

Paso 1: Configurar una cuenta de AWS y crear un usuario

Antes de usar Amazon Rekognition por primera vez, debe completar las tareas siguientes:

1. Regístrate para obtener una AWS cuenta.
2. Crear un usuario.

En esta sección de la guía para desarrolladores, se explica por qué y cómo va a crear una cuenta de AWS y un usuario.

Temas

- Crea una AWS cuenta y un usuario

Crea una AWS cuenta y un usuario

Cuentas de AWS

Cuando se inscribe en Amazon Web Services (AWS), la cuenta de AWS se registra automáticamente en todos los servicios de AWS, incluido Amazon Rekognition. Solo se le cobrará por los servicios que utilice.

Con Amazon Rekognition, paga solo por los recursos que usa.

Si es un AWS cliente nuevo, puede empezar a utilizar Amazon Rekognition de forma gratuita. Para obtener más información, consulte Capa gratuita de AWS.

Consulte la próxima sección, Inscríbase en un Cuenta de AWS, para ver las instrucciones de creación de cuentas.

Si ya tiene una AWS cuenta, omita la configuración de la cuenta y cree un usuario administrativo.

Usuarios

Para tener acceso a los servicios de AWS, como Amazon Rekognition, debe proporcionar credenciales. Esto es así para que el servicio puede determinar si usted tiene permisos para obtener acceso a los recursos que son propiedad de tal servicio.

Puede crear claves de acceso para su AWS cuenta para acceder a la consola AWS CLI o, APIs mientras usa la consola, necesitará su contraseña. Sin embargo, no es recomendable que acceda a AWS con las credenciales de usuario raíz de su cuenta de AWS. En su lugar, le recomendamos que utilice AWS Identity and Access Management (IAM) para crear un usuario administrativo.

A continuación, podrá obtener acceso a AWS mediante una dirección URL especial y esas credenciales de usuario administrativo.

Si se ha inscrito en AWS pero no ha creado un usuario, puede crear uno mediante la consola de IAM. Consulte la siguiente sección, Creación de un usuario con acceso administrativo, para obtener instrucciones sobre cómo crear un usuario administrativo.

Inscríbase en un 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 Consola de administración de AWScomo 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: Configure y AWS CLI AWS SDKs

Temas

- Concesión de acceso programático

- Uso de Rekognition con un SDK AWS

Los siguientes pasos muestran cómo instalar el AWS Command Line Interface (AWS CLI) y qué AWS SDKs se utilizan en los ejemplos de esta documentación. Hay varias formas diferentes de autenticar las llamadas al AWS SDK. En los ejemplos de esta guía se parte del supuesto de que utilizas un perfil de credenciales predeterminado para llamar a AWS CLI los comandos y a las operaciones de la API AWS del SDK.

Para obtener una lista de AWS las regiones disponibles, consulta Regiones y puntos finales en la Referencia general de Amazon Web Services.

Siga los pasos para descargar y configurar la AWS SDKs.

Para configurar el AWS CLI y el AWS SDKs

1. Descargue e instale el AWS CLIy el AWS SDKs que desee usar. Esta guía proporciona ejemplos de Java AWS CLI, Python, Ruby, Node.js, PHP, .NET y JavaScript. Para obtener información sobre la instalación AWS SDKs, consulte Herramientas para Amazon Web Services.
2. Cree una clave de acceso para el usuario que ha creado en Crea una AWS cuenta y un usuario.
   a. Inicie sesión en la consola de IAM Consola de administración de AWS y ábrala en https://console.aws.amazon.com/iam/.
   b. En el panel de navegación, seleccione Usuarios.
   c. Elija el nombre del usuario que ha cread en Crea una AWS cuenta y un usuario.
   d. Seleccione la pestaña de credenciales de seguridad.
   e. Elija Create access key (Crear clave de acceso). A continuación, elija Download .csv file (Descargar archivo .csv) para guardar el ID de clave de acceso y la clave de acceso secreta en un archivo CSV de su equipo. Guarde el archivo en un lugar seguro. No podrá obtener acceso de nuevo a la clave de acceso secreta cuando este cuadro de diálogo se cierre. Cuando haya acabado de descargar el archivo CSV, seleccione Close.
3. Si ha instalado la AWS CLI, puede configurar las credenciales y la región de la mayoría AWS SDKs introduciéndolas `aws configure` en la línea de comandos. De lo contrario, utilice las instrucciones siguientes.
4. En su equipo, vaya al directorio principal y cree un directorio `.aws`. En sistemas basados en Unix, como Linux o macOS, se encuentra en la ubicación siguiente:

```
~/.aws
```

En Windows, se encuentra en la siguiente ubicación:

```
%HOMEPATH%\.aws
```

5. En el directorio `.aws`, cree un archivo denominado `credentials`.
6. Abra el archivo CSV de credenciales que creó en el paso 2 y copie su contenido en el archivo `credentials` con el formato siguiente:

```
[default]
aws_access_key_id = your_access_key_id
aws_secret_access_key = your_secret_access_key
```

Especifique su ID de clave de acceso y la clave de acceso secreta en lugar de your_access_key_id y your_secret_access_key.

7. Guarde el archivo `Credentials` y elimine el archivo CSV.
8. En el directorio `.aws`, cree un archivo denominado `config`.
9. Abra el archivo `config` e introduzca su región con el formato siguiente.

```
[default]
region = your_aws_region
```

Reemplace su_región_de_aws por la región de AWS que desee (por ejemplo `us-west-2`).

> **Note**
>
> Si no selecciona una región, se usará us-east-1 de forma predeterminada.

10. Guarde el archivo `config`.

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. Consola de administración de AWS 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
IAM	(Recomendado) Utilice las credenciales de la consola como 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 Iniciar sesión para el desarrollo AWS local en la Guía del AWS Command Line Interface usuario. • Para ello AWS SDKs, consulte Iniciar sesión para el desarrollo AWS local en la Guía de referencia de AWS SDKs and Tools.

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

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

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 `profile-name`. Por ejemplo:

```
session = boto3.Session(profile_name="profile-name")
rekognition_client = session.client("rekognition")
```

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

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 `profile-name`.

- 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: utilice el proveedor de credenciales de punto de conexión de metadatos de la instancia de Amazon EC2. El proveedor generará y actualizará automáticamente las credenciales por usted mediante el perfil de instancia de Amazon EC2 que asocie a la instancia de Amazon

EC2. Para obtener más información, consulte Uso de un rol de IAM para conceder permisos a aplicaciones que se ejecutan en instancias de Amazon EC2.

- Amazon Elastic Container Service: utilice el proveedor de credenciales de Container. Amazon ECS envía y actualiza las credenciales a un punto de conexión de metadatos. Un rol de IAM de tarea que indique proporcionará una estrategia para administrar las credenciales que utilice su aplicación. Para obtener más información, consulte Interacción con servicios de AWS.

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

Uso de Rekognition con un SDK AWS

AWS Los kits de desarrollo de software (SDKs) están disponibles para muchos lenguajes de programación populares. Cada SDK proporciona una API, ejemplos de código y documentación que facilitan a los desarrolladores la creación de aplicaciones en su lenguaje preferido.

Documentación de SDK	Ejemplos de código
AWS SDK para C++	AWS SDK para C++ ejemplos de código
AWS CLI	AWS CLI ejemplos de código
AWS SDK para Go	AWS SDK para Go ejemplos de código
AWS SDK para Java	AWS SDK para Java ejemplos de código
AWS SDK para JavaScript	AWS SDK para JavaScript ejemplos de código
AWS SDK para Kotlin	AWS SDK para Kotlin ejemplos de código
AWS SDK para .NET	AWS SDK para .NET ejemplos de código
AWS SDK para PHP	AWS SDK para PHP ejemplos de código
Herramientas de AWS para PowerShell	Herramientas de AWS para PowerShell ejemplos de código
AWS SDK para Python (Boto3)	AWS SDK para Python (Boto3) ejemplos de código

Documentación de SDK	Ejemplos de código
AWS SDK para Ruby	AWS SDK para Ruby ejemplos de código
AWS SDK para Rust	AWS SDK para Rust ejemplos de código
AWS SDK para SAP ABAP	AWS SDK para SAP ABAP ejemplos de código
AWS SDK para Swift	AWS SDK para Swift ejemplos de código

Para ver ejemplos específicos de Rekognition, consulte Ejemplos de código para Amazon Rekognition mediante AWS SDKs.

> **Ejemplo de disponibilidad**
>
> ¿No encuentra lo que necesita? Solicite un ejemplo de código a través del enlace de Enviar comentarios que se encuentra al final de esta página.

Paso 3: Cómo empezar a usar la API AWS CLI y el AWS SDK

Una vez que haya configurado el AWS CLI y AWS SDKs que desee usar, puede crear aplicaciones que usen Amazon Rekognition. La mayoría de las interacciones con Amazon Rekognition se producen mediante las operaciones de la API, si bien algunas características concretas de este tienen flujos de trabajo de consola.

En los siguientes temas se muestra cómo empezar a utilizar Amazon Rekognition Image y Amazon Rekognition Video a través de o el. AWS CLI AWS SDKs

- Trabajar con imágenes: cubre el proceso de análisis de imágenes con Amazon Rekognition Image.
- Cómo trabajar con operaciones de análisis de vídeos almacenados: cubre el proceso de análisis de vídeo almacenado sin transmisión con Amazon Rekognition Video.
- Trabajar con eventos de vídeo en streaming: cubre el proceso de análisis de transmisión de vídeo con Amazon Rekognition Video.

En las secciones enumeradas anteriormente se incluyen ejemplos en los que se utiliza la AWS CLI. Si piensa utilizar el AWS CLI, consulte la siguiente sección para obtener información sobre cómo formatear las llamadas a la API.

Formatear los AWS CLI ejemplos

Los AWS CLI ejemplos de esta guía están formateados para el sistema operativo Linux. Para utilizar los ejemplos con Microsoft Windows, deberá cambiar el formato JSON del parámetro `--image`, así como los saltos de línea de barras diagonales inversas (\) por signos de intercalación (^). Para obtener más información sobre el formato JSON, consulte Especificación de valores de parámetros para la AWS Command Line Interface.

El siguiente es un ejemplo de AWS CLI comando formateado para Microsoft Windows (tenga en cuenta que estos comandos no se ejecutarán tal cual, son solo ejemplos de formato):

```
aws rekognition detect-labels ^
  --image "{\"S3Object\":{\"Bucket\":\"photo-collection\",\"Name\":\"photo.jpg\"}}" ^
  --region region-name
```

También puede proporcionar una versión abreviada de JSON que funcione en Microsoft Windows y Linux.

```
aws rekognition detect-labels --image "S3Object={Bucket=photo-
collection,Name=photo.jpg}" --region region-name
```

Para obtener más información, consulte Uso de sintaxis abreviada con la AWS Command Line Interface.

Siguiente paso

Paso 4: Introducción al uso de la consola de Amazon Rekognition

Paso 4: Introducción al uso de la consola de Amazon Rekognition

La consola de Amazon Rekognition permite administrar los recursos relacionados con las características Etiquetas personalizadas y Moderación personalizada de Rekognition. La consola solo ofrece demostraciones de otras características de Rekognition.

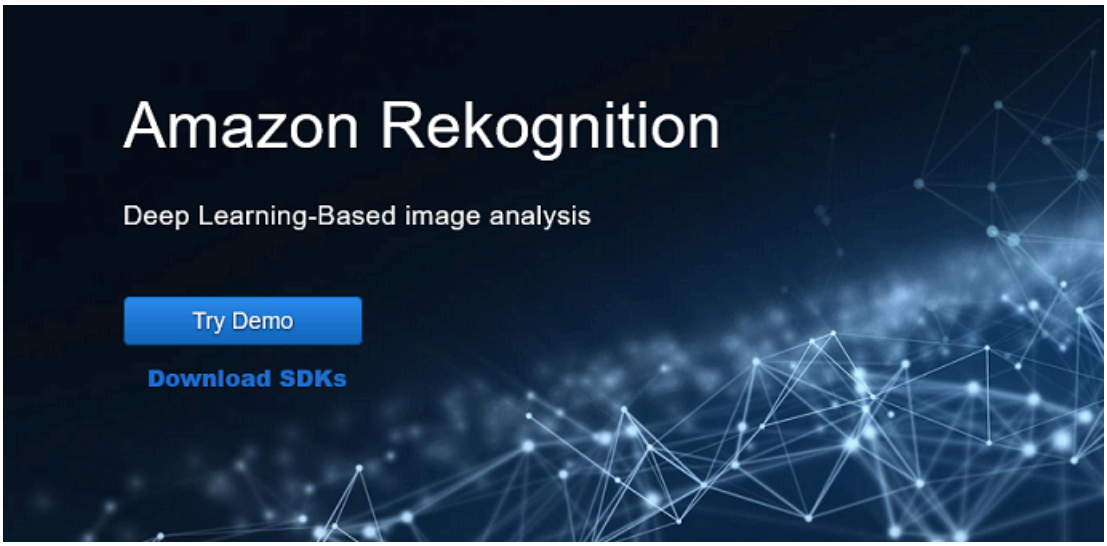
Esta sección muestra cómo utilizar un subconjunto de las capacidades de Amazon Rekognition como la detección de objetos y escenas, el análisis facial y la comparación de rostros en un conjunto de

imágenes. Para obtener más información, consulte Cómo funciona Amazon Rekognition. También puede usar la AWS CLI API Amazon Rekognition para detectar objetos y escenas, detectar rostros y comparar y buscar rostros. Para obtener más información, consulte Paso 3: Cómo empezar a usar la API AWS CLI y el AWS SDK.

En esta sección también se muestra cómo ver las CloudWatch métricas agregadas de Amazon para Rekognition mediante la consola de Rekognition.

Temas

- Configurar los permisos de la consola
- Ejercicio 1: Detectar objetos y escenas (consola)
- Ejercicio 2: Analizar rostros de una imagen (consola)
- Ejercicio 3: Comparar rostros en imágenes (consola)
- Ejercicio 4: Consultar métricas totales (consola)



Configurar los permisos de la consola

Para usar la consola de Rekognition, debe tener los permisos adecuados para el rol o la cuenta que accede a la consola. Para algunas operaciones, Rekognition creará automáticamente un bucket de Amazon S3 para almacenar los archivos administrados durante la operación. Si desea almacenar sus archivos de entrenamiento en un bucket diferente a este bucket de consola, necesitará permisos adicionales.

Permiso de acceso a la consola

Para usar la consola Rekognition, puede usar una política de IAM como la siguiente, que cubre Amazon S3 y la consola de Rekognition. Para conocer cómo asignar permisos, consulte Asignación de permisos.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "RekognitionFullAccess",
            "Effect": "Allow",
            "Action": [
                "rekognition:*"
            ],
            "Resource": "*"
        },
        {
            "Sid": "RekognitionConsoleS3BucketSearchAccess",
            "Effect": "Allow",
            "Action": [
                "s3:ListAllMyBuckets",
                "s3:ListBucket",
                "s3:GetBucketAcl",
                "s3:GetBucketLocation"
            ],
            "Resource": "*"
        },
        {
            "Sid": "RekognitionConsoleS3BucketFirstUseSetupAccess",
            "Effect": "Allow",
            "Action": [
                "s3:CreateBucket",
                "s3:PutBucketVersioning",
                "s3:PutLifecycleConfiguration",
                "s3:PutEncryptionConfiguration",
                "s3:PutBucketPublicAccessBlock",
                "s3:PutBucketCors",
                "s3:GetBucketCors"
            ],
```

```
            "Resource": "arn:aws:s3:::rekognition-custom-projects-*"
        },
        {
            "Sid": "RekognitionConsoleS3BucketAccess",
            "Effect": "Allow",
            "Action": [
                "s3:ListBucket",
                "s3:GetBucketLocation",
                "s3:GetBucketVersioning"
            ],
            "Resource": "arn:aws:s3:::rekognition-custom-projects-*"
        },
        {
            "Sid": "RekognitionConsoleS3ObjectAccess",
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:DeleteObject",
                "s3:GetObjectAcl",
                "s3:GetObjectTagging",
                "s3:GetObjectVersion",
                "s3:PutObject"
            ],
            "Resource": "arn:aws:s3:::rekognition-custom-projects-*/*"
        },
        {
            "Sid": "RekognitionConsoleManifestAccess",
            "Effect": "Allow",
            "Action": [
                "groundtruthlabeling:*"
            ],
            "Resource": "*"
        },
        {
            "Sid": "RekognitionConsoleTagSelectorAccess",
            "Effect": "Allow",
            "Action": [
                "tag:GetTagKeys",
                "tag:GetTagValues"
            ],
            "Resource": "*"
        },
        {
            "Sid": "RekognitionConsoleKmsKeySelectorAccess",
```

```
            "Effect": "Allow",
            "Action": [
                "kms:ListAliases"
            ],
            "Resource": "*"
        }
    ]
}
```

Acceso a buckets de Amazon S3 externos

Cuando abres la consola de Rekognition por primera vez en una AWS región nueva, Rekognition crea un depósito (depósito de consola) que se utiliza para almacenar los archivos del proyecto. También puede utilizar su propio bucket de Amazon S3 (bucket externo) para subir las imágenes o el archivo de manifiesto en la consola. Para usar un bucket externo, agregue el siguiente bloque de políticas a la política anterior.

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

Asignación de permisos

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

- Usuarios y grupos en AWS IAM Identity Center (sucesor de AWS Single Sign-On):

  Cree un conjunto de permisos. Siga las instrucciones de Create a permission set en la Guía del usuario de AWS IAM Identity Center (sucesor de AWS Single Sign-On).

- 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.

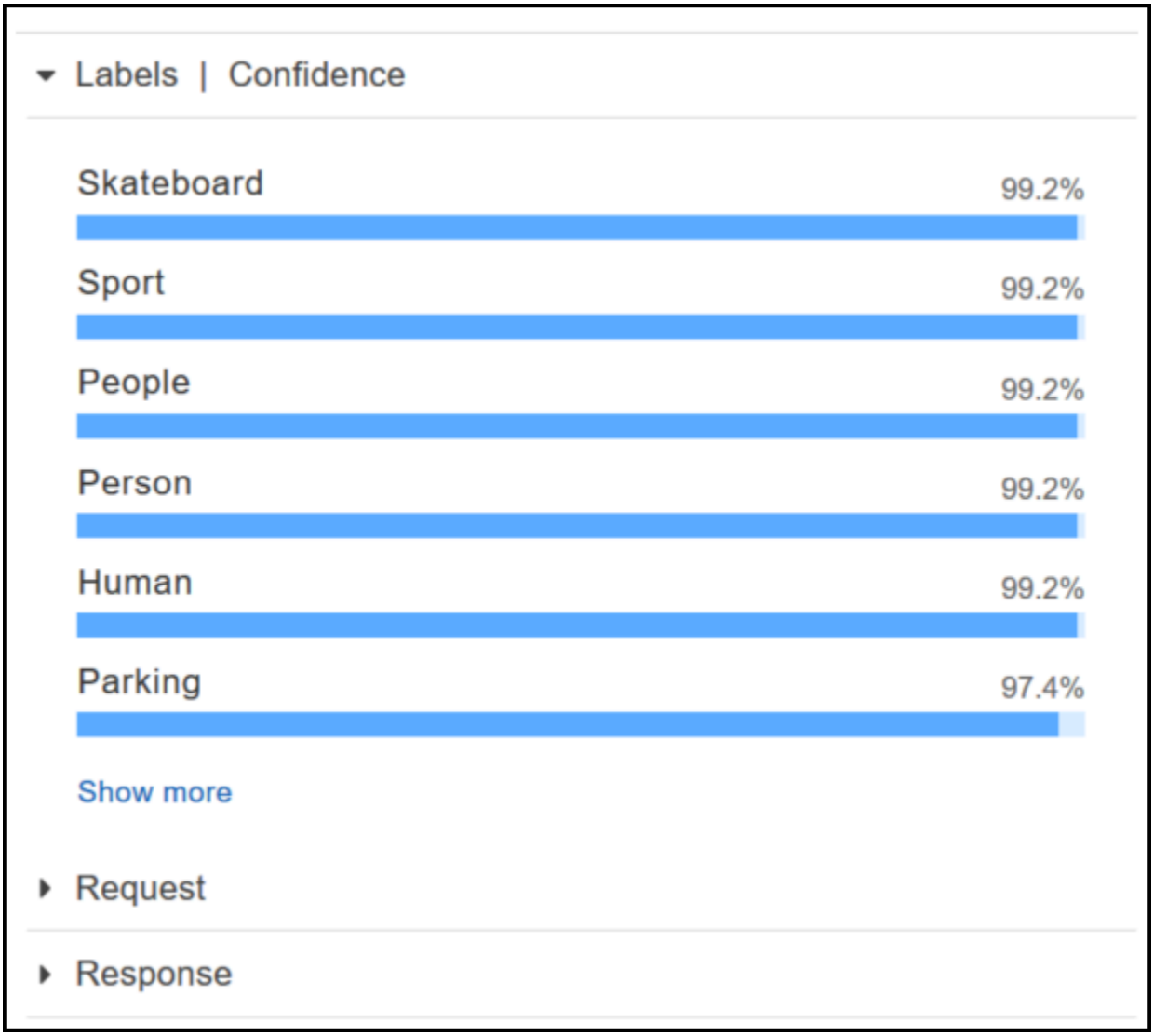
Ejercicio 1: Detectar objetos y escenas (consola)

En esta sección se explica de manera muy general cómo funciona la capacidad de detección de objetos y escenas de Amazon Rekognition. Cuando especifica una imagen como entrada, el servicio detecta los objetos y las escenas de la imagen y los devuelve junto con una puntuación de porcentaje de confianza en forma de porcentaje de cada objeto y escena.

Por ejemplo, Amazon Rekognition detecta los siguientes objetos y escenas en la imagen de muestra: skateboard, deporte, persona, auto, coche y vehículo.



Amazon Rekognition devuelve también una puntuación de confianza para cada objeto detectado en la imagen de ejemplo, como se muestra en la siguiente respuesta de ejemplo.



Para ver todas las puntuaciones de confianza que se muestran en esta respuesta, elija Show more (Mostrar más) en el panel Labels | Confidence (Etiquetas | Confianza).

También puede examinar la solicitud a la API y la respuesta de la API como referencia.

Solicitud

```
{
   "contentString":{
      "Attributes":[
         "ALL"
      ],
      "Image":{
         "S3Object":{
            "Bucket":"console-sample-images",
```

```
            "Name":"skateboard.jpg"
         }
      }
   }
}
```

Respuesta

```
{
   "Labels":[
      {
         "Confidence":99.25359344482422,
         "Name":"Skateboard"
      },
      {
         "Confidence":99.25359344482422,
         "Name":"Sport"
      },
      {
         "Confidence":99.24723052978516,
         "Name":"People"
      },
      {
         "Confidence":99.24723052978516,
         "Name":"Person"
      },
      {
         "Confidence":99.23908233642578,
         "Name":"Human"
      },
      {
         "Confidence":97.42484283447266,
         "Name":"Parking"
      },
      {
         "Confidence":97.42484283447266,
         "Name":"Parking Lot"
      },
      {
         "Confidence":91.53300476074219,
         "Name":"Automobile"
      },
      {
```

```
            "Confidence":91.53300476074219,
            "Name":"Car"
         },
         {
            "Confidence":91.53300476074219,
            "Name":"Vehicle"
         },
         {
            "Confidence":76.85114288330078,
            "Name":"Intersection"
         },
         {
            "Confidence":76.85114288330078,
            "Name":"Road"
         },
         {
            "Confidence":76.21503448486328,
            "Name":"Boardwalk"
         },
         {
            "Confidence":76.21503448486328,
            "Name":"Path"
         },
         {
            "Confidence":76.21503448486328,
            "Name":"Pavement"
         },
         {
            "Confidence":76.21503448486328,
            "Name":"Sidewalk"
         },
         {
            "Confidence":76.21503448486328,
            "Name":"Walkway"
         },
         {
            "Confidence":66.71541595458984,
            "Name":"Building"
         },
         {
            "Confidence":62.04711151123047,
            "Name":"Coupe"
         },
         {
```

```
            "Confidence":62.04711151123047,
            "Name":"Sports Car"
         },
         {
            "Confidence":61.98909378051758,
            "Name":"City"
         },
         {
            "Confidence":61.98909378051758,
            "Name":"Downtown"
         },
         {
            "Confidence":61.98909378051758,
            "Name":"Urban"
         },
         {
            "Confidence":60.978023529052734,
            "Name":"Neighborhood"
         },
         {
            "Confidence":60.978023529052734,
            "Name":"Town"
         },
         {
            "Confidence":59.22066116333008,
            "Name":"Sedan"
         },
         {
            "Confidence":56.48063278198242,
            "Name":"Street"
         },
         {
            "Confidence":54.235477447509766,
            "Name":"Housing"
         },
         {
            "Confidence":53.85226058959961,
            "Name":"Metropolis"
         },
         {
            "Confidence":52.001792907714844,
            "Name":"Office Building"
         },
         {
```

```
            "Confidence":51.325313568115234,
            "Name":"Suv"
         },
         {
            "Confidence":51.26075744628906,
            "Name":"Apartment Building"
         },
         {
            "Confidence":51.26075744628906,
            "Name":"High Rise"
         },
         {
            "Confidence":50.68067932128906,
            "Name":"Pedestrian"
         },
         {
            "Confidence":50.59548568725586,
            "Name":"Freeway"
         },
         {
            "Confidence":50.568580627441406,
            "Name":"Bumper"
         }
      ]
}
```


Para obtener más información, consulte Cómo funciona Amazon Rekognition.

Detectar objetos y escenas en una imagen proporcionada por el usuario

Puede subir una imagen de su propiedad o proporcionar la dirección URL de una imagen como entrada en la consola de Amazon Rekognition. Amazon Rekognition devuelve el objeto y las escenas y las puntuaciones de confianza de cada objeto y escena que detecta en la imagen proporcionada.

> **Note**
>
> La imagen debe tener menos de 5 MB y debe estar en formato JPEG o PNG.

Para detectar objetos y escenas en una imagen proporcionada por el usuario

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

2. Elija Detección de etiquetas.
3. Realice una de las siguientes acciones:
   - Suba una imagen: elija Upload, vaya a la ubicación donde guardó la imagen y selecciónela.
   - Use una dirección URL: escriba la dirección URL en el cuadro de texto y elija Go.
4. Consulte la puntuación de confianza de cada etiqueta detectada en el panel Labels | Confidence.

Para obtener más opciones de análisis de imágenes, consulte the section called “Trabajar con imágenes”.

Detecte personas y objetos en un vídeo que proporcione

Puede subir un vídeo que proporciona como entrada en la consola de Amazon Rekognition. Amazon Rekognition devuelve las personas, los objetos y las etiquetas detectadas en el vídeo.

> **Note**
>
> El vídeo de demostración no debe tener más de un minuto de duración ni más de 30 MB. Debe estar en formato de MP4 archivo y estar codificado con el códec H.264.

Para detectar personas y objetos en un vídeo que proporcione

1. Abra la consola Amazon Rekognition en. https://console.aws.amazon.com/rekognition/
2. Elija Análisis de vídeos almacenados en la barra de navegación.
3. En Elegir una muestra o cargar la suya propia, seleccione Su propio vídeo en el menú desplegable.
4. Arrastre y suelte el vídeo o selecciónelo en la ubicación en la que lo guardó.

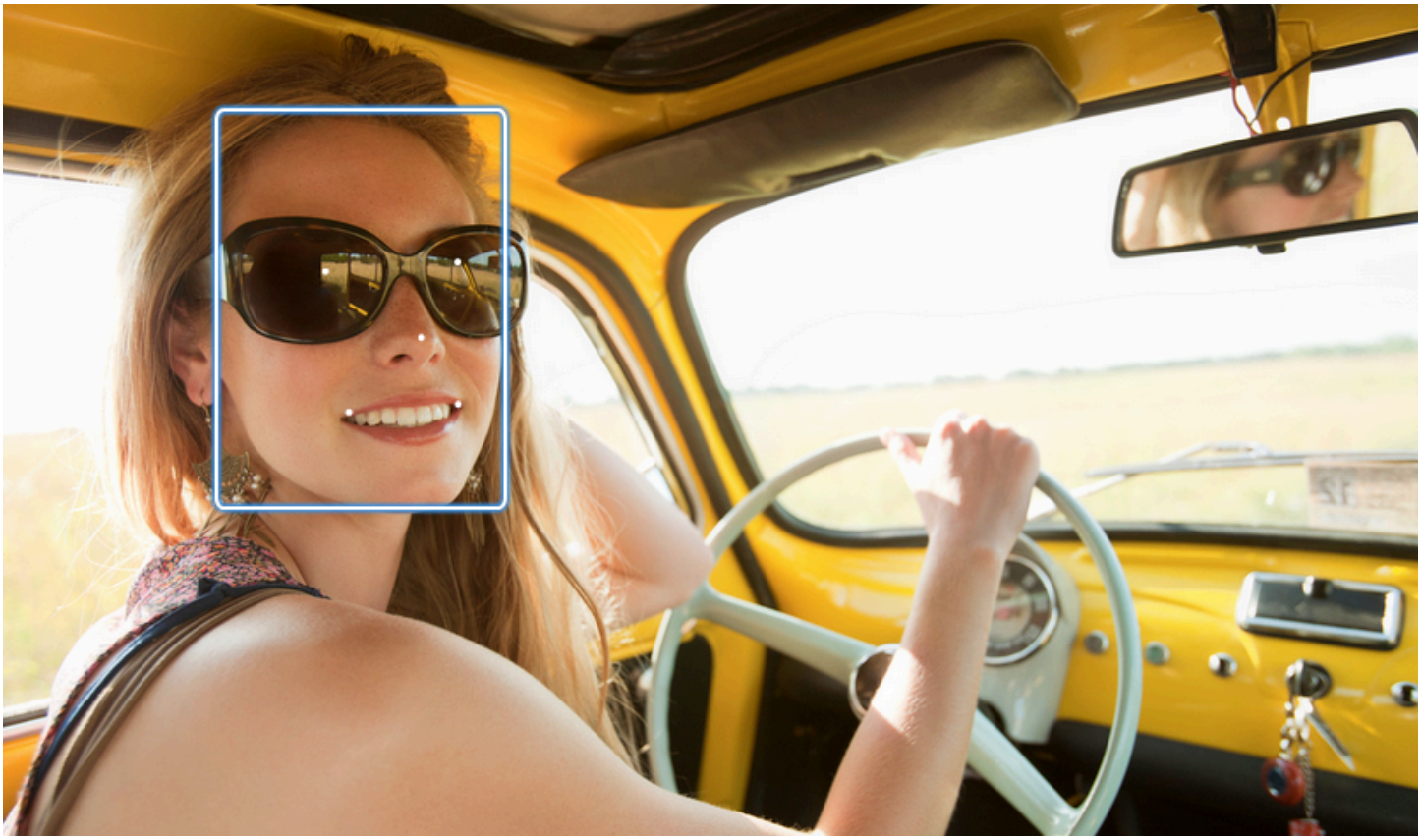
Para obtener más opciones de análisis de vídeo, consulte the section called “Cómo trabajar con operaciones de análisis de vídeos almacenados” o the section called “Trabajar con eventos de vídeo en streaming”.

Ejercicio 2: Analizar rostros de una imagen (consola)

Esta sección muestra cómo utilizar la consola de Amazon Rekognition para detectar rostros y analizar atributos faciales en una imagen. Cuando se proporciona una imagen que contiene un rostro

como entrada, el servicio detecta el rostro en la imagen, analiza los atributos faciales y devuelve una puntuación de confianza en forma de porcentaje para el rostro y los atributos faciales detectados en la imagen. Para obtener más información, consulte Cómo funciona Amazon Rekognition.

Por ejemplo, si elige la siguiente imagen de muestra como entrada, Amazon Rekognition la detecta como un rostro y devuelve puntuaciones de confianza para el rostro y los atributos faciales detectados.



A continuación se muestra la respuesta de ejemplo.

Results



looks like a face	99.8%
appears to be female	100%
age range	23 - 38 years old
smiling	99.4%
appears to be happy	93.2%
wearing eyeglasses	99.9%
wearing sunglasses	97.6%
eyes are open	96.2%
mouth is open	72.5%
does not have a mustache	77.6%
does not have a beard	97.1%

Show less

Si hay varios rostros en la imagen de entrada, Rekognition detecta hasta 100 rostros en la imagen. Cada rostro detectado se marca con un cuadrado. Al hacer clic en el área marcada con un cuadrado en un rostro, Rekognition muestra la puntuación de confianza de ese rostro y sus atributos detectados en el panel Rostros | Confianza.

Analizar rostros de una imagen proporcionada por el usuario

Puede subir su propia imagen o proporcionar la dirección URL de la imagen en la consola de Amazon Rekognition.

> **Note**
>
> La imagen debe tener menos de 5 MB y debe estar en formato JPEG o PNG.

Para analizar un rostro en una imagen proporcionada por el usuario

1. Abra la consola Amazon Rekognition en. https://console.aws.amazon.com/rekognition/
2. Elija Facial analysis.
3. Realice una de las siguientes acciones:
   - Suba una imagen: elija Upload, vaya a la ubicación donde guardó la imagen y selecciónela.
   - Use una dirección URL: escriba la dirección URL en el cuadro de texto y elija Go.
4. Consulte la puntuación de confianza de uno de los rostros detectados y sus atributos faciales en el panel Faces | Confidence.
5. Si hay varios rostros en la imagen, elija uno de los demás rostros para ver sus atributos y puntuaciones.

Ejercicio 3: Comparar rostros en imágenes (consola)

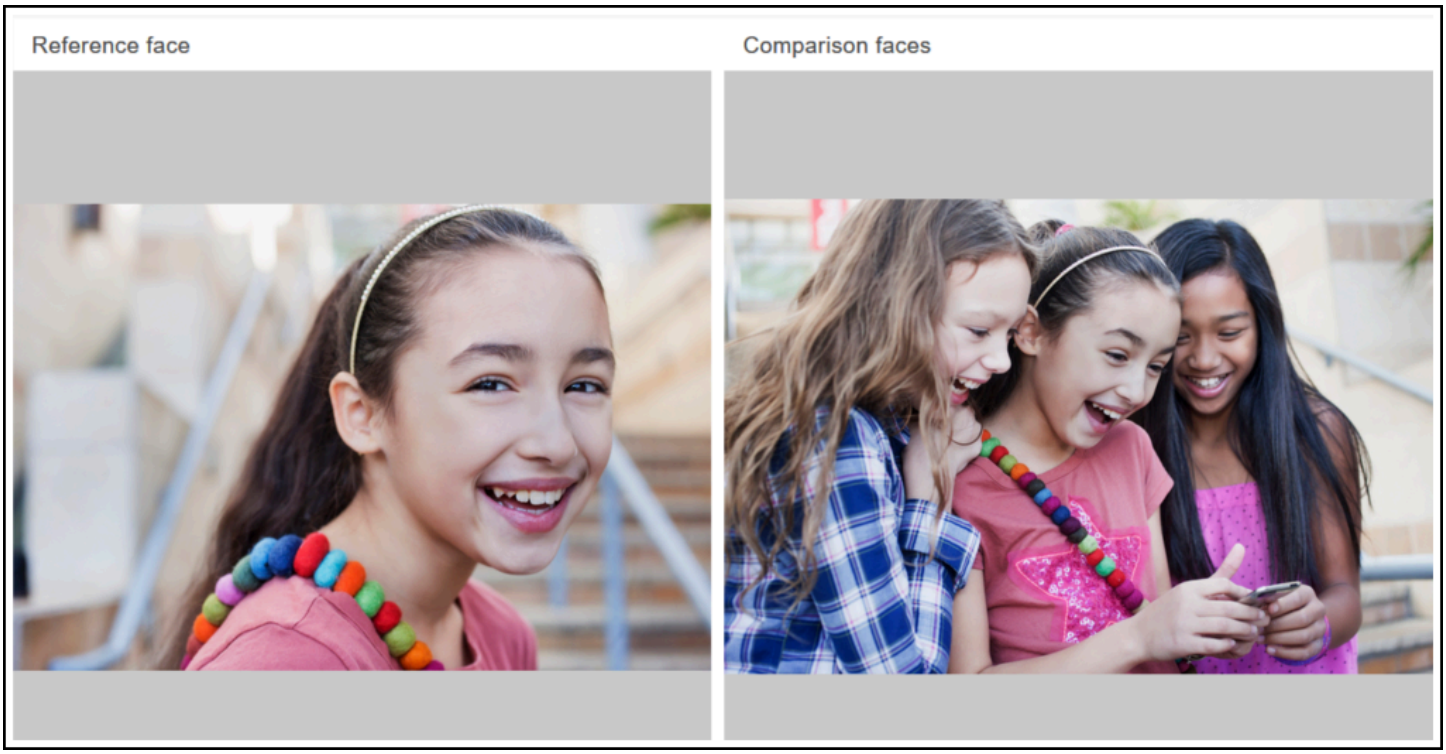
Esta sección muestra cómo utilizar la consola de Amazon Rekognition para comparar rostros en un conjunto de imágenes que contienen varios rostros. Cuando se especifica una imagen Rostro de referencia (origen) y una imagen Rostros de comparación (destino), Rekognition compara el rostro de mayor tamaño de la imagen de origen (es decir, el rostro de referencia) con hasta 100 rostros detectados en la imagen de destino (es decir, los rostros de comparación) y determina la similitud del rostro de la imagen de origen con los rostros de la imagen de destino. La puntuación de similitud de cada comparación se muestra en el panel Results.

Si la imagen de destino contiene varios rostros, Rekognition compara el rostro de la imagen de origen con hasta 100 rostros detectados en la imagen de destino y asigna una puntuación de similitud a cada comparación.

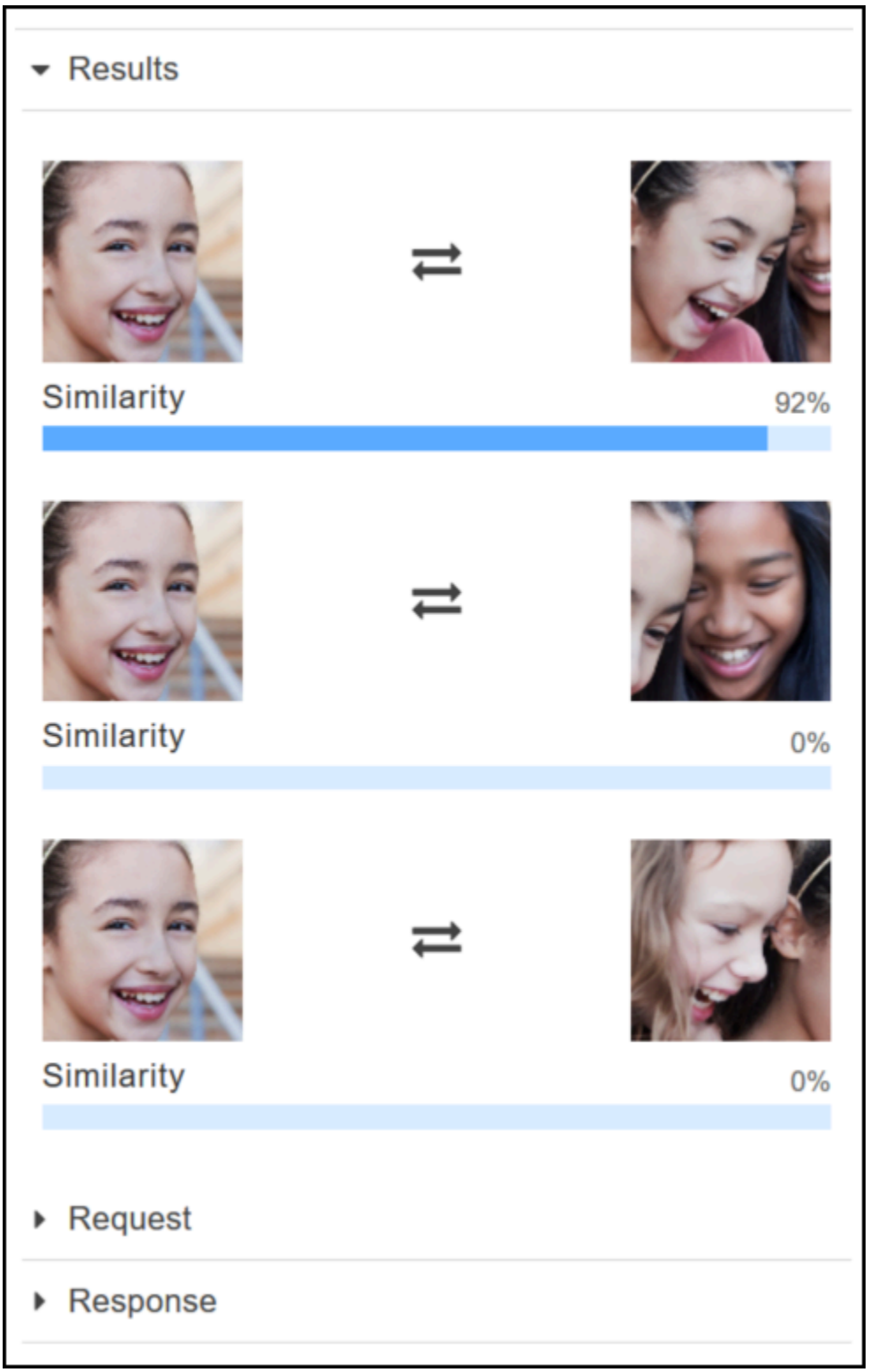
Si la imagen de origen contiene varios rostros, el servicio detecta el rostro más grande en la imagen de origen y lo usa para compararlo con cada rostro detectado en la imagen de destino.

Para obtener más información, consulte Comparación de rostros en imágenes.

Por ejemplo, con la imagen de ejemplo que se muestra a la izquierda como imagen de origen y la imagen de ejemplo que se muestra a la derecha como imagen de destino, Rekognition detecta el rostro de la imagen de origen, lo compara con cada rostro de la imagen de destino y muestra una puntuación de similitud para cada par.



A continuación, se muestran los rostros detectados en la imagen de destino y la puntuación de similitud de cada rostro.



Comparar rostros de una imagen proporcionada por el usuario

Puede subir sus propias imágenes de origen y de destino para que Rekognition compare los rostros de las imágenes o puede especificar una URL para la ubicación de las imágenes.

> **Note**
>
> La imagen debe tener menos de 5 MB y debe estar en formato JPEG o PNG.

Para comparar rostros en las imágenes

1. Abra la consola Amazon Rekognition en. https://console.aws.amazon.com/rekognition/
2. Elija Face comparison.
3. Para la imagen de origen, realice alguna de las siguientes operaciones:
   - Suba una imagen: elija Upload a la izquierda, vaya a la ubicación donde guardó su imagen de origen y, a continuación, seleccione la imagen.
   - Use una dirección URL: escriba la dirección URL de la imagen de origen en el cuadro de texto y elija Go.
4. Para la imagen de destino, realice alguna de las siguientes operaciones:
   - Suba una imagen: elija Upload a la derecha, vaya a la ubicación donde guardó su imagen de origen y, a continuación, seleccione la imagen.
   - Use una dirección URL: escriba la dirección URL de la imagen de origen en el cuadro de texto y elija Go.
5. Rekognition compara el rostro de mayor tamaño de la imagen de origen con hasta 100 rostros de la imagen de destino y muestra la puntuación de similitud de cada pareja en el panel Resultados.

Ejercicio 4: Consultar métricas totales (consola)

El panel de métricas de Amazon Rekognition muestra gráficos de actividad para el total acumulado de las distintas métricas de Rekognition durante un periodo de tiempo especificado. Por ejemplo, la métrica global `SuccessfulRequestCount` muestra el número total de solicitudes realizadas con éxito a todas las operaciones de API de Rekognition durante los últimos siete días.

En la siguiente tabla se indican los gráficos que se muestra en el panel de métricas de Rekognition y la métrica de Rekognition correspondiente. Para obtener más información, consulte CloudWatch métricas de Rekognition.

Gráfico	Métrica global
Llamadas realizadas correctamente	SuccessfulRequestCount
Errores del cliente	UserErrorCount
Errores del servidor	ServerErrorCount
Solicitudes restringidas	ThrottledCount
Etiquetas detectadas	DetectedLabelCount
Rostros detectados	DetectedFaceCount

Cada gráfico muestra los datos de las métricas globales durante un periodo de tiempo determinado. Se muestra también el número total de datos de métricas globales durante el periodo de tiempo. Para ver las métricas de llamadas a la API individuales, elija el enlace situado debajo de cada gráfico.

Para permitir que los usuarios accedan al panel de métricas de Rekognition, asegúrese de que el usuario tenga los permisos de Rekognition adecuados. CloudWatch Por ejemplo, un usuario con los permisos de política administrados `AmazonRekognitionReadOnlyAccess` y `CloudWatchReadOnlyAccess` puede ver el panel de las métricas. Si un usuario no tiene los permisos necesarios, cuando el usuario abra el panel de métricas, no aparecerá ningún gráfico. Para obtener más información, consulte Administración de identidades y accesos para Amazon Rekognition.

Para obtener más información sobre la supervisión de Rekognition CloudWatch con, consulte. Monitorización de Rekognition con Amazon CloudWatch

Para ver métricas globales (consola)

1. Abra la consola Amazon Rekognition en. https://console.aws.amazon.com/rekognition/
2. En el panel de navegación, seleccione Métricas.
3. En el menú desplegable, seleccione el periodo de tiempo durante el que desea obtener métricas.

4. Para actualizar los gráficos, seleccione el botón Refresh.
5. Para ver CloudWatch las métricas detalladas de una métrica agregada específica, seleccione Ver detalles CloudWatch debajo del gráfico de métricas.

Trabajar con imágenes y vídeos

Puede utilizar las operaciones de la API de Amazon Rekognition con tres tipos distintos de contenido multimedia: imágenes, vídeos almacenados y transmisiones de vídeo. En esta sección se proporciona información general sobre cómo escribir código que accede a Amazon Rekognition para procesar los distintos tipos de contenido multimedia. Para obtener instrucciones acerca de las prácticas recomendadas y las consideraciones que deben tenerse en cuenta, consulte las secciones correspondientes que aparecen a continuación en función del tipo de contenido multimedia que procese.

En otras secciones de esta guía se proporciona información sobre tipos específicos de análisis de imágenes y vídeo, como la detección de rostros.

Temas

- Trabajar con imágenes
- Cómo trabajar con operaciones de análisis de vídeos almacenados
- Trabajar con eventos de vídeo en streaming
- Gestión de errores
- Utilizar Amazon Rekognition como servicio autorizado de FedRAMP

Trabajar con imágenes

En esta sección, se describen los tipos de análisis que Amazon Rekognition Image puede realizar en las imágenes.

- Detección de objetos y escenas
- Detección y comparación de rostros
- Búsqueda de rostros en una colección
- Reconocimiento de famosos
- Moderación de imágenes
- Detección de texto en una imagen

Se llevan a cabo mediante operaciones de API sin almacenamiento en las que Amazon Rekognition Image no conserva la información detectada por la operación. Las operaciones API sin

almacenamiento no conservan bytes de imagen de entrada. Para obtener más información, consulte Descripción de operaciones de API con almacenamiento y sin almacenamiento.

Amazon Rekognition Image también puede almacenar metadatos faciales en colecciones para recuperación posterior. Para obtener más información, consulte Búsqueda de rostros en una colección.

En esta sección se utilizan las operaciones de API de Amazon Rekognition Image para analizar imágenes almacenadas en un bucket de Amazon S3 y bytes de imágenes subidos desde el sistema de archivos local. En esta sección también se explica cómo obtener información de orientación de imagen a partir de una imagen .jpg.

Rekognition solo usa canales RGB para realizar inferencias. AWS recomienda a los usuarios eliminar el canal alfa antes de utilizar una pantalla para inspeccionar visualmente (manualmente por una persona) la comparación.

Temas

- Especificaciones de imagen
- Análisis de imágenes almacenadas en un bucket de Amazon S3
- Análisis de una imagen subida desde un sistema de archivos local
- Visualización de cuadros delimitadores
- Obtención de coordenadas de cuadro delimitador y orientación de imagen

Especificaciones de imagen

Las operaciones de Amazon Rekognition Image pueden analizar imágenes en formato .jpg o .png.

Puede pasar bytes de imágenes a una operación de Amazon Rekognition Image como parte de la llamada o hacer referencia a un objeto de S3 existente. Para un ejemplo de análisis de una imagen almacenada en un bucket de Amazon S3, consulte Análisis de imágenes almacenadas en un bucket de Amazon S3. Para ver un ejemplo del paso de bytes de imagen a una operación API en Amazon Rekognition Image, consulte Análisis de una imagen subida desde un sistema de archivos local.

Si utiliza HTTP y transfiere los bytes de imagen como parte de una operación de Amazon Rekognition Image, dichos bytes deben pasarse como una cadena codificada en base64. Si utiliza el AWS SDK y transfiere los bytes de imagen como parte de la llamada a la operación API, el requisito de codificar en base64 los bytes de la imagen dependerá del lenguaje que utilice.

Las siguientes opciones comunes codifican AWS SDKs automáticamente en base64 las imágenes y no es necesario codificar los bytes de las imágenes antes de llamar a una operación de la API Amazon Rekognition Image.

- Java
- JavaScript
- Python
- PHP

Si utiliza otro AWS SDK y obtiene un error de formato de imagen al llamar a una operación API de Rekognition;, pruebe el cifrado en base64 de los bytes de imagen antes de transferirlos a una operación API de Rekognition.

Si utilizas el AWS CLI para llamar a las operaciones de Amazon Rekognition Image, no se admite el paso de bytes de imagen como parte de la llamada. Debe subir primero la imagen en un bucket de Amazon S3 y, a continuación, llamar a la operación que hace referencia a la imagen subida.

> **Note**
>
> No es necesario que la imagen esté cifrada en base64 si transfiere una imagen almacenada en un `S3Object` en lugar de bytes de imagen.

Para obtener información acerca de garantizar la mínima latencia posible para operaciones de Amazon Rekognition Image, consulte Latencia de operación de Amazon Rekognition Image.

Corrección de la orientación de imagen

En varias operaciones API de Rekognition, se devuelve la orientación de una imagen analizada. Conocer la orientación de imagen es importante, ya que le permite reorientar las imágenes para su visualización. Las operaciones de API de Rekognition que analizan rostros también devuelven cuadros delimitadores para la ubicación de rostros dentro de una imagen. Puede utilizar los cuadros delimitadores para mostrar un recuadro alrededor de un rostro en una imagen. Las coordenadas del cuadro delimitador devueltas se ven afectadas por la orientación de la imagen y es posible que tenga que traducir las coordenadas del cuadro delimitador para mostrar correctamente un cuadro alrededor de un rostro. Para obtener más información, consulte Obtención de coordenadas de cuadro delimitador y orientación de imagen.

Redimensionamiento de imagen

Durante el análisis, Amazon Rekognition cambia el tamaño interno de las imágenes mediante un conjunto de rangos predefinidos que mejor se adaptan a un modelo o algoritmo en particular. Por este motivo, Amazon Rekognition puede detectar un número diferente de objetos o proporcionar resultados distintos, en función de la resolución de la imagen de entrada. Por ejemplo, supongamos que tiene dos imágenes. La primera imagen tiene una resolución de 1024 x 768 píxeles. La segunda imagen, una versión redimensionada de la primera imagen, tiene una resolución de 640 x 480 píxeles. Si envía las imágenes a DetectLabels, las respuestas de las dos llamadas `DetectLabels` pueden diferir ligeramente.

Análisis de imágenes almacenadas en un bucket de Amazon S3

Amazon Rekognition Image puede analizar imágenes almacenadas en un bucket de Amazon S3 o imágenes suministradas como bytes de imagen.

En este tema, utilizará la operación de DetectLabelsAPI para detectar objetos, conceptos y escenas en una imagen (JPEG o PNG) almacenada en un bucket de Amazon S3. Puede transferir una imagen a una operación API de Amazon Rekognition Image mediante el parámetro de entrada Image. Dentro de `Image`, especifique la propiedad de objeto S3Object para hacer referencia a una imagen almacenada en un bucket de S3. Los bytes de imagen para las imágenes almacenadas en buckets de Amazon S3 no tienen por qué estar codificados en base64. Para obtener más información, consulte Especificaciones de imagen.

Ejemplo de solicitud

En este ejemplo de solicitud de JSON para `DetectLabels`, la imagen de origen (`input.jpg`) se sube desde un bucket de Amazon S3; llamado `amzn-s3-demo-bucket`. La región del bucket de Amazon S3 que contiene el objeto S3 debe coincidir con la región que utiliza para las operaciones de Amazon Rekognition Image.

```
{
    "Image": {
        "S3Object": {
            "Bucket": "amzn-s3-demo-bucket",
            "Name": "input.jpg"
        }
    },
    "MaxLabels": 10,
    "MinConfidence": 75
```

```
}
```

En los ejemplos siguientes se utilizan varios AWS SDKs y la llamada AWS CLI to`DetectLabels`. Para obtener más información sobre la respuesta de la operación `DetectLabels`, consulte DetectLabels respuesta.

Para detectar las etiquetas en una imagen

1. Si aún no lo ha hecho:

   a. Cree o actualice un usuario con los permisos `AmazonRekognitionFullAccess` y `AmazonS3ReadOnlyAccess`. Para obtener más información, consulte Paso 1: Configurar una cuenta de AWS y crear un usuario.

   b. Instale y configure el AWS CLI y el AWS SDKs. Para obtener más información, consulte Paso 2: Configure y AWS CLI AWS SDKs. Asegúrese de haber otorgado al usuario que llama a las operaciones de la API los permisos adecuados para el acceso mediante programación; consulte las instrucciones de Concesión de acceso programático sobre cómo hacerlo.

2. Suba una imagen que contenga uno o varios objetos, como árboles, casas o barcos, en el bucket de S3. La imagen debe estar en formato .jpg o .png.

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

3. Consulte los siguientes ejemplos para llamar a la operación `DetectLabels`.

   Java

   Este ejemplo muestra una lista de las etiquetas que se han detectado en la imagen de entrada. Reemplace los valores de `bucket` y `photo` por los nombre del bucket de Amazon S3 y de la imagen que utilizó en el paso 2.

```
//Copyright 2018 Amazon.com, Inc. or its affiliates. All Rights Reserved.
//PDX-License-Identifier: MIT-0 (For details, see https://github.com/awsdocs/
amazon-rekognition-developer-guide/blob/master/LICENSE-SAMPLECODE.)

package com.amazonaws.samples;
import com.amazonaws.services.rekognition.AmazonRekognition;
import com.amazonaws.services.rekognition.AmazonRekognitionClientBuilder;
import com.amazonaws.services.rekognition.model.AmazonRekognitionException;
```

```
import com.amazonaws.services.rekognition.model.DetectLabelsRequest;
import com.amazonaws.services.rekognition.model.DetectLabelsResult;
import com.amazonaws.services.rekognition.model.Image;
import com.amazonaws.services.rekognition.model.Label;
import com.amazonaws.services.rekognition.model.S3Object;
import java.util.List;

public class DetectLabels {

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

      String photo = "input.jpg";
      String bucket = "bucket";


      AmazonRekognition rekognitionClient =
 AmazonRekognitionClientBuilder.defaultClient();

      DetectLabelsRequest request = new DetectLabelsRequest()
           .withImage(new Image()
           .withS3Object(new S3Object()
           .withName(photo).withBucket(bucket)))
           .withMaxLabels(10)
           .withMinConfidence(75F);

      try {
         DetectLabelsResult result = rekognitionClient.detectLabels(request);
         List <Label> labels = result.getLabels();

         System.out.println("Detected labels for " + photo);
         for (Label label: labels) {
            System.out.println(label.getName() + ": " +
 label.getConfidence().toString());
         }
      } catch(AmazonRekognitionException e) {
         e.printStackTrace();
      }
   }
}
```

AWS CLI

Este ejemplo muestra la salida de JSON de la operación `detect-labels` de la CLI. Reemplace los valores de `bucket` y `photo` por los nombre del bucket de Amazon S3 y de la imagen que utilizó en el paso 2. Sustituya el valor de `profile_name` en la línea que crea la sesión de Rekognition por el nombre de su perfil de desarrollador.

```
aws rekognition detect-labels --image '{ "S3Object": { "Bucket": "bucket-name",
 "Name": "file-name" } }' \
--features GENERAL_LABELS IMAGE_PROPERTIES \
--settings '{"ImageProperties": {"MaxDominantColors":1}, {"GeneralLabels":
{"LabelInclusionFilters":["Cat"]}}}' \
--profile profile-name \
--region us-east-1
```

Si utiliza Windows, es posible que tenga que evitar las comillas, como se muestra en el siguiente ejemplo.

```
aws rekognition detect-labels --image "{\"S3Object\":{\"Bucket\":\"bucket-
name\",\"Name\":\"file-name\"}}" --features GENERAL_LABELS IMAGE_PROPERTIES --
settings "{\"GeneralLabels\":{\"LabelInclusionFilters\":[\"Car\"]}}" --profile
 profile-name --region us-east-1
```

Java V2

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

```
//snippet-start:[rekognition.java2.detect_labels.import]
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.Image;
import software.amazon.awssdk.services.rekognition.model.DetectLabelsRequest;
import software.amazon.awssdk.services.rekognition.model.DetectLabelsResponse;
import software.amazon.awssdk.services.rekognition.model.Label;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import software.amazon.awssdk.services.rekognition.model.S3Object;
import java.util.List;
```

```
/**
* Before running this Java V2 code example, set up your development environment,
 including your credentials.
*
* For more information, see the following documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
*/
public class DetectLabels {

    public static void main(String[] args) {

        final String usage = "\n" +
            "Usage: " +
            "   <bucket> <image>\n\n" +
            "Where:\n" +
            "   bucket - The name of the Amazon S3 bucket that contains the
 image (for example, ,ImageBucket)." +
            "   image - The name of the image located in the Amazon S3 bucket
 (for example, Lake.png). \n\n";

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucket = args[0];
        String image = args[1];
        Region region = Region.US_WEST_2;
        RekognitionClient rekClient = RekognitionClient.builder()
            .region(region)
            .credentialsProvider(ProfileCredentialsProvider.create("profile-
name"))
            .build();

        getLabelsfromImage(rekClient, bucket, image);
        rekClient.close();
    }

    // snippet-start:[rekognition.java2.detect_labels_s3.main]
    public static void getLabelsfromImage(RekognitionClient rekClient, String
 bucket, String image) {
```

```
        try {
            S3Object s3Object = S3Object.builder()
                .bucket(bucket)
                .name(image)
                .build() ;

            Image myImage = Image.builder()
                .s3Object(s3Object)
                .build();

            DetectLabelsRequest detectLabelsRequest =
 DetectLabelsRequest.builder()
                .image(myImage)
                .maxLabels(10)
                .build();

            DetectLabelsResponse labelsResponse =
 rekClient.detectLabels(detectLabelsRequest);
            List<Label> labels = labelsResponse.labels();
            System.out.println("Detected labels for the given photo");
            for (Label label: labels) {
                System.out.println(label.name() + ": " +
 label.confidence().toString());
            }

        } catch (RekognitionException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
 // snippet-end:[rekognition.java2.detect_labels.main]
}
```

Python

Este ejemplo muestra las etiquetas que se han detectado en la imagen de entrada. Reemplace los valores de `bucket` y `photo` por los nombre del bucket de Amazon S3 y de la imagen que utilizó en el paso 2. Sustituya el valor de `profile_name` en la línea que crea la sesión de Rekognition por el nombre de su perfil de desarrollador.

```
#Copyright 2018 Amazon.com, Inc. or its affiliates. All Rights Reserved.
#PDX-License-Identifier: MIT-0 (For details, see https://github.com/awsdocs/
amazon-rekognition-developer-guide/blob/master/LICENSE-SAMPLECODE.)
```

```
import boto3

def detect_labels(photo, bucket):

     session = boto3.Session(profile_name='profile-name')
     client = session.client('rekognition')

     response = client.detect_labels(Image={'S3Object':
{'Bucket':bucket,'Name':photo}},
     MaxLabels=10,
     # Uncomment to use image properties and filtration settings
     #Features=["GENERAL_LABELS", "IMAGE_PROPERTIES"],
     #Settings={"GeneralLabels": {"LabelInclusionFilters":["Cat"]},
     # "ImageProperties": {"MaxDominantColors":10}}
     )

     print('Detected labels for ' + photo)
     print()
     for label in response['Labels']:
         print("Label: " + label['Name'])
         print("Confidence: " + str(label['Confidence']))
         print("Instances:")

         for instance in label['Instances']:
             print(" Bounding box")
             print(" Top: " + str(instance['BoundingBox']['Top']))
             print(" Left: " + str(instance['BoundingBox']['Left']))
             print(" Width: " + str(instance['BoundingBox']['Width']))
             print(" Height: " + str(instance['BoundingBox']['Height']))
             print(" Confidence: " + str(instance['Confidence']))
             print()

         print("Parents:")
         for parent in label['Parents']:
            print(" " + parent['Name'])

         print("Aliases:")
         for alias in label['Aliases']:
             print(" " + alias['Name'])

             print("Categories:")
         for category in label['Categories']:
             print(" " + category['Name'])
```

```
            print("----------")
            print()

    if "ImageProperties" in str(response):
        print("Background:")
        print(response["ImageProperties"]["Background"])
        print()
        print("Foreground:")
        print(response["ImageProperties"]["Foreground"])
        print()
        print("Quality:")
        print(response["ImageProperties"]["Quality"])
        print()

    return len(response['Labels'])

def main():
    photo = 'photo-name'
    bucket = 'amzn-s3-demo-bucket'
    label_count = detect_labels(photo, bucket)
    print("Labels detected: " + str(label_count))

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

Node.Js

En este ejemplo se muestra información acerca de las etiquetas que se detectan en una imagen.

Cambie el valor de `photo` por la ruta y el nombre de un archivo de imagen que contenga uno o más rostros de famosos. Cambie el valor de `bucket` por el nombre del bucket de S3 que contiene el archivo de imagen proporcionado. Sustituya el valor de `REGION` por el nombre de la región asociada a su cuenta. Sustituya el valor de `profile_name` en la línea que crea la sesión de Rekognition por el nombre de su perfil de desarrollador.

```
// Import required AWS SDK clients and commands for Node.js
import { DetectLabelsCommand } from  "@aws-sdk/client-rekognition";
import  { RekognitionClient } from "@aws-sdk/client-rekognition";

import {fromIni} from '@aws-sdk/credential-providers';
```

```
// Set the AWS Region.
const REGION = "region-name"; //e.g. "us-east-1"

// Create SNS service object.
const rekogClient = new RekognitionClient({
  region: REGION,
  credentials: fromIni({
       profile: 'profile-name',
  }),
});

const bucket = 'bucket-name'
const photo = 'photo-name'

// Set params
const params = {For example, to grant
    Image: {
      S3Object: {
        Bucket: bucket,
        Name: photo
      },
    },
  }

const detect_labels = async () => {
    try {
        const response = await rekogClient.send(new
 DetectLabelsCommand(params));
        console.log(response.Labels)
        response.Labels.forEach(label =>{
            console.log(`Confidence: ${label.Confidence}`)
            console.log(`Name: ${label.Name}`)
            console.log('Instances:')
            label.Instances.forEach(instance => {
                console.log(instance)
            })
            console.log('Parents:')
            label.Parents.forEach(name => {
                console.log(name)
            })
            console.log("-------")
        })
        return response; // For unit tests.
      } catch (err) {
```

```
        console.log("Error", err);
      }
};

detect_labels();
```

.NET

Este ejemplo muestra una lista de las etiquetas que se han detectado en la imagen de entrada. Reemplace los valores de `bucket` y `photo` por los nombre del bucket de Amazon S3 y de la imagen que utilizó en el paso 2.

```
//Copyright 2018 Amazon.com, Inc. or its affiliates. All Rights Reserved.
//PDX-License-Identifier: MIT-0 (For details, see https://github.com/awsdocs/
amazon-rekognition-developer-guide/blob/master/LICENSE-SAMPLECODE.)

using System;
using Amazon.Rekognition;
using Amazon.Rekognition.Model;

public class DetectLabels
{
    public static void Example()
    {
        String photo = "input.jpg";
        String bucket = "amzn-s3-demo-bucket";

        AmazonRekognitionClient rekognitionClient = new
 AmazonRekognitionClient();

        DetectLabelsRequest detectlabelsRequest = new DetectLabelsRequest()
        {
            Image = new Image()
            {
                S3Object = new S3Object()
                {
                    Name = photo,
                    Bucket = bucket
                },
            },
            MaxLabels = 10,
            MinConfidence = 75F
        };
```

```
        try
        {
            DetectLabelsResponse detectLabelsResponse =
 rekognitionClient.DetectLabels(detectlabelsRequest);
            Console.WriteLine("Detected labels for " + photo);
            foreach (Label label in detectLabelsResponse.Labels)
                Console.WriteLine("{0}: {1}", label.Name, label.Confidence);
        }
        catch (Exception e)
        {
            Console.WriteLine(e.Message);
        }
    }
}
```

Ruby

Este ejemplo muestra una lista de las etiquetas que se han detectado en la imagen de entrada. Reemplace los valores de `bucket` y `photo` por los nombre del bucket de Amazon S3 y de la imagen que utilizó en el paso 2.

```
  # Add to your Gemfile
  # gem 'aws-sdk-rekognition'
  require 'aws-sdk-rekognition'
  credentials = Aws::Credentials.new(
     ENV['AWS_ACCESS_KEY_ID'],
     ENV['AWS_SECRET_ACCESS_KEY']
  )
  bucket = 'bucket' # the bucket name without s3://
  photo  = 'photo' # the name of file
  client   = Aws::Rekognition::Client.new credentials: credentials
  attrs = {
    image: {
      s3_object: {
        bucket: bucket,
        name: photo
      },
    },
    max_labels: 10
  }
```

```
response = client.detect_labels attrs
puts "Detected labels for: #{photo}"
response.labels.each do |label|
  puts "Label:      #{label.name}"
  puts "Confidence: #{label.confidence}"
  puts "Instances:"
  label['instances'].each do |instance|
    box = instance['bounding_box']
    puts "  Bounding box:"
    puts "    Top:        #{box.top}"
    puts "    Left:       #{box.left}"
    puts "    Width:      #{box.width}"
    puts "    Height:     #{box.height}"
    puts "  Confidence: #{instance.confidence}"
  end
  puts "Parents:"
  label.parents.each do |parent|
    puts "  #{parent.name}"
  end
  puts "------------"
  puts ""
end
```

Ejemplo de respuesta

La respuesta de `DetectLabels` es una matriz de las etiquetas detectadas en la imagen y el nivel de confianza por el que se detectan.

Al realizar la operación `DetectLabels` en una imagen, Amazon Rekognition devuelve un resultado similar al siguiente ejemplo de respuesta.

La respuesta muestra que la operación ha detectado varias etiquetas, como Person (Persona), Vehicle (Vehículo) y Car (Automóvil). Cada etiqueta tiene un nivel de confianza asociado. Por ejemplo, el algoritmo de detección tiene una confianza del 98,991432 % en que la imagen contiene una persona.

La respuesta también incluye las etiquetas antecesoras de una etiqueta de la matriz `Parents`. Por ejemplo, la etiqueta Automobile (Automóvil) tiene dos etiquetas principales denominadas Vehicle (Vehículo) y Transportation (Transporte).

La respuesta para las etiquetas de objetos comunes incluye información del cuadro delimitador para localizar la etiqueta en la imagen de entrada. Por ejemplo, la etiqueta Person (persona) tiene una matriz de instancias que contiene dos cuadros delimitadores. Se trata de las ubicaciones de dos personas detectadas en la imagen.

El campo `LabelModelVersion` contiene el número de versión del modelo de detección que `DetectLabels` utiliza.

Para obtener más información acerca del uso de la operación `DetectLabels`, consulte Detección de objetos y conceptos.


```
{

    {
    "Labels": [
        {
            "Name": "Vehicle",
            "Confidence": 99.15271759033203,
            "Instances": [],
            "Parents": [
                {
                    "Name": "Transportation"
                }
            ]
        },
        {
            "Name": "Transportation",
            "Confidence": 99.15271759033203,
            "Instances": [],
            "Parents": []
        },
        {
            "Name": "Automobile",
            "Confidence": 99.15271759033203,
            "Instances": [],
            "Parents": [
                {
                    "Name": "Vehicle"
                },
                {
                    "Name": "Transportation"
                }
            ]
```

```
        },
        {
            "Name": "Car",
            "Confidence": 99.15271759033203,
            "Instances": [
                {
                    "BoundingBox": {
                        "Width": 0.10616336017847061,
                        "Height": 0.18528179824352264,
                        "Left": 0.0037978808395564556,
                        "Top": 0.5039216876029968
                    },
                    "Confidence": 99.15271759033203
                },
                {
                    "BoundingBox": {
                        "Width": 0.2429988533258438,
                        "Height": 0.21577216684818268,
                        "Left": 0.7309805154800415,
                        "Top": 0.5251884460449219
                    },
                    "Confidence": 99.1286392211914
                },
                {
                    "BoundingBox": {
                        "Width": 0.14233611524105072,
                        "Height": 0.15528248250484467,
                        "Left": 0.6494812965393066,
                        "Top": 0.5333095788955688
                    },
                    "Confidence": 98.48368072509766
                },
                {
                    "BoundingBox": {
                        "Width": 0.11086395382881165,
                        "Height": 0.10271988064050674,
                        "Left": 0.10355594009160995,
                        "Top": 0.5354844927787781
                    },
                    "Confidence": 96.45606231689453
                },
                {
                    "BoundingBox": {
                        "Width": 0.06254628300666809,
```

```
                "Height": 0.053911514580249786,
                "Left": 0.46083059906959534,
                "Top": 0.5573825240135193
            },
            "Confidence": 93.65448760986328
        },
        {
            "BoundingBox": {
                "Width": 0.10105438530445099,
                "Height": 0.12226245552301407,
                "Left": 0.5743985772132874,
                "Top": 0.534368634223938
            },
            "Confidence": 93.06217193603516
        },
        {
            "BoundingBox": {
                "Width": 0.056389667093753815,
                "Height": 0.17163699865341187,
                "Left": 0.9427769780158997,
                "Top": 0.5235804319381714
            },
            "Confidence": 92.6864013671875
        },
        {
            "BoundingBox": {
                "Width": 0.06003860384225845,
                "Height": 0.06737709045410156,
                "Left": 0.22409997880458832,
                "Top": 0.5441341400146484
            },
            "Confidence": 90.4227066040039
        },
        {
            "BoundingBox": {
                "Width": 0.02848697081208229,
                "Height": 0.19150497019290924,
                "Left": 0.0,
                "Top": 0.5107086896896362
            },
            "Confidence": 86.65286254882812
        },
        {
            "BoundingBox": {
```

```
                    "Width": 0.04067881405353546,
                    "Height": 0.03428703173995018,
                    "Left": 0.316415935754776,
                    "Top": 0.5566273927688599
                },
                "Confidence": 85.36471557617188
            },
            {
                "BoundingBox": {
                    "Width": 0.043411049991846085,
                    "Height": 0.0893595889210701,
                    "Left": 0.18293385207653046,
                    "Top": 0.5394920110702515
                },
                "Confidence": 82.21705627441406
            },
            {
                "BoundingBox": {
                    "Width": 0.031183116137981415,
                    "Height": 0.03989990055561066,
                    "Left": 0.2853088080883026,
                    "Top": 0.5579366683959961
                },
                "Confidence": 81.0157470703125
            },
            {
                "BoundingBox": {
                    "Width": 0.031113790348172188,
                    "Height": 0.056484755128622055,
                    "Left": 0.2580395042896271,
                    "Top": 0.5504819750785828
                },
                "Confidence": 56.13441467285156
            },
            {
                "BoundingBox": {
                    "Width": 0.08586374670267105,
                    "Height": 0.08550430089235306,
                    "Left": 0.5128012895584106,
                    "Top": 0.5438792705535889
                },
                "Confidence": 52.37760925292969
            }
        ],
```

```
            "Parents": [
                {
                    "Name": "Vehicle"
                },
                {
                    "Name": "Transportation"
                }
            ]
        },
        {
            "Name": "Human",
            "Confidence": 98.9914321899414,
            "Instances": [],
            "Parents": []
        },
        {
            "Name": "Person",
            "Confidence": 98.9914321899414,
            "Instances": [
                {
                    "BoundingBox": {
                        "Width": 0.19360728561878204,
                        "Height": 0.2742200493812561,
                        "Left": 0.43734854459762573,
                        "Top": 0.35072067379951477
                    },
                    "Confidence": 98.9914321899414
                },
                {
                    "BoundingBox": {
                        "Width": 0.03801717236638069,
                        "Height": 0.06597328186035156,
                        "Left": 0.9155802130699158,
                        "Top": 0.5010883808135986
                    },
                    "Confidence": 85.02790832519531
                }
            ],
            "Parents": []
        }
    ],
    "LabelModelVersion": "2.0"
}
```

```
}
```

Análisis de una imagen subida desde un sistema de archivos local

Las operaciones de Amazon Rekognition Image pueden analizar imágenes suministradas como bytes de imagen o almacenadas en un bucket de Amazon S3.

En estos temas se ofrecen ejemplos sobre el suministro de bytes de imágenes a operaciones de API de Amazon Rekognition Image mediante un archivo subido a partir de un sistema de archivos local. Puede transferir bytes de imágenes a una operación API de Amazon Rekognition; utilizando el parámetro de entrada Image. Dentro de `Image`, especifique la propiedad `Bytes` para transferir bytes de imágenes con codificación en base64.

Los bytes de imagen transferidos a una operación API de Amazon Rekognition; utilizando el parámetro de entrada `Bytes` deben estar cifrados en base64. La AWS SDKs que utilizan estos ejemplos codifica automáticamente las imágenes en base64. No es necesario codificar bytes de imágenes antes de llamar a una operación API de Amazon Rekognition. Para obtener más información, consulte Especificaciones de imagen.

En esta solicitud de ejemplo de JSON para `DetectLabels`, los bytes de imagen de origen se pasan al parámetro de entrada `Bytes`.

```
{
    "Image": {
        "Bytes": "/9j/4AAQSk....."
    },
    "MaxLabels": 10,
    "MinConfidence": 77
}
```

En los siguientes ejemplos se utilizan varios AWS SDKs y el to call. AWS CLI `DetectLabels` Para obtener más información sobre la respuesta de la operación `DetectLabels`, consulte DetectLabels respuesta.

Para ver un JavaScript ejemplo del lado del cliente, consulte. Usando JavaScript

Para detectar las etiquetas en una imagen local

1. Si aún no lo ha hecho:

a. Cree o actualice un usuario con los permisos `AmazonRekognitionFullAccess` y `AmazonS3ReadOnlyAccess`. Para obtener más información, consulte Paso 1: Configurar una cuenta de AWS y crear un usuario.

b. Instale y configure el AWS CLI y el. AWS SDKs Para obtener más información, consulte Paso 2: Configure y AWS CLI AWS SDKs.

2. Consulte los siguientes ejemplos para llamar a la operación `DetectLabels`.

Java

El siguiente ejemplo de Java muestra cómo subir una imagen desde el sistema de archivos local y detectar etiquetas mediante la operación detectLabels del SDK de AWS. Cambie el valor de `photo` por la ruta y el nombre de un archivo de imagen (en formato .jpg o .png).

```
//Copyright 2018 Amazon.com, Inc. or its affiliates. All Rights Reserved.
//PDX-License-Identifier: MIT-0 (For details, see https://github.com/awsdocs/
amazon-rekognition-developer-guide/blob/master/LICENSE-SAMPLECODE.)

package aws.example.rekognition.image;
import java.io.File;
import java.io.FileInputStream;
import java.io.InputStream;
import java.nio.ByteBuffer;
import java.util.List;
import com.amazonaws.services.rekognition.AmazonRekognition;
import com.amazonaws.services.rekognition.AmazonRekognitionClientBuilder;
import com.amazonaws.AmazonClientException;
import com.amazonaws.services.rekognition.model.AmazonRekognitionException;
import com.amazonaws.services.rekognition.model.DetectLabelsRequest;
import com.amazonaws.services.rekognition.model.DetectLabelsResult;
import com.amazonaws.services.rekognition.model.Image;
import com.amazonaws.services.rekognition.model.Label;
import com.amazonaws.util.IOUtils;

public class DetectLabelsLocalFile {
    public static void main(String[] args) throws Exception {
     String photo="input.jpg";


        ByteBuffer imageBytes;
        try (InputStream inputStream = new FileInputStream(new File(photo))) {
            imageBytes = ByteBuffer.wrap(IOUtils.toByteArray(inputStream));
```

```
        }


        AmazonRekognition rekognitionClient =
 AmazonRekognitionClientBuilder.defaultClient();

        DetectLabelsRequest request = new DetectLabelsRequest()
                .withImage(new Image()
                        .withBytes(imageBytes))
                .withMaxLabels(10)
                .withMinConfidence(77F);

        try {

            DetectLabelsResult result =
 rekognitionClient.detectLabels(request);
            List <Label> labels = result.getLabels();

            System.out.println("Detected labels for " + photo);
            for (Label label: labels) {
               System.out.println(label.getName() + ": " +
 label.getConfidence().toString());
            }

        } catch (AmazonRekognitionException e) {
            e.printStackTrace();
        }

    }
}
```

Python

El siguiente ejemplo de AWS SDK for Python muestra cómo subir una imagen desde el sistema de archivos local y llamar a la operación detect_labels. Cambie el valor de `photo` por la ruta y el nombre de un archivo de imagen (en formato .jpg o .png).

```
#Copyright 2018 Amazon.com, Inc. or its affiliates. All Rights Reserved.
#PDX-License-Identifier: MIT-0 (For details, see https://github.com/awsdocs/
amazon-rekognition-developer-guide/blob/master/LICENSE-SAMPLECODE.)

import boto3
```

```
def detect_labels_local_file(photo):


    client=boto3.client('rekognition')

    with open(photo, 'rb') as image:
        response = client.detect_labels(Image={'Bytes': image.read()})

    print('Detected labels in ' + photo)
    for label in response['Labels']:
        print (label['Name'] + ' : ' + str(label['Confidence']))

    return len(response['Labels'])

def main():
    photo='photo'

    label_count=detect_labels_local_file(photo)
    print("Labels detected: " + str(label_count))


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

.NET

El siguiente ejemplo de muestra cómo subir una imagen desde el sistema de archivos local y detectar etiquetas utilizando la operación `DetectLabels`. Cambie el valor de `photo` por la ruta y el nombre de un archivo de imagen (en formato .jpg o .png).

```
//Copyright 2018 Amazon.com, Inc. or its affiliates. All Rights Reserved.
//PDX-License-Identifier: MIT-0 (For details, see https://github.com/awsdocs/
amazon-rekognition-developer-guide/blob/master/LICENSE-SAMPLECODE.)

using System;
using System.IO;
using Amazon.Rekognition;
using Amazon.Rekognition.Model;
```

```
public class DetectLabelsLocalfile
{
    public static void Example()
    {
        String photo = "input.jpg";

        Amazon.Rekognition.Model.Image image = new
 Amazon.Rekognition.Model.Image();
        try
        {
            using (FileStream fs = new FileStream(photo, FileMode.Open,
 FileAccess.Read))
            {
                byte[] data = null;
                data = new byte[fs.Length];
                fs.Read(data, 0, (int)fs.Length);
                image.Bytes = new MemoryStream(data);
            }
        }
        catch (Exception)
        {
            Console.WriteLine("Failed to load file " + photo);
            return;
        }

        AmazonRekognitionClient rekognitionClient = new
 AmazonRekognitionClient();

        DetectLabelsRequest detectlabelsRequest = new DetectLabelsRequest()
        {
            Image = image,
            MaxLabels = 10,
            MinConfidence = 77F
        };

        try
        {
            DetectLabelsResponse detectLabelsResponse =
 rekognitionClient.DetectLabels(detectlabelsRequest);
            Console.WriteLine("Detected labels for " + photo);
            foreach (Label label in detectLabelsResponse.Labels)
                Console.WriteLine("{0}: {1}", label.Name, label.Confidence);
        }
        catch (Exception e)
```

```
        {
            Console.WriteLine(e.Message);
        }
    }
}
```

PHP

El siguiente ejemplo de AWS SDK for PHP muestra cómo cargar una imagen desde el sistema de archivos local y llamar a la operación DetectFacesAPI. Cambie el valor de `photo` por la ruta y el nombre de un archivo de imagen (en formato .jpg o .png).

```
<?php
//Copyright 2018 Amazon.com, Inc. or its affiliates. All Rights Reserved.
//PDX-License-Identifier: MIT-0 (For details, see https://github.com/awsdocs/
amazon-rekognition-developer-guide/blob/master/LICENSE-SAMPLECODE.)

    require 'vendor/autoload.php';

    use Aws\Rekognition\RekognitionClient;

    $options = [
       'region'            => 'us-west-2',
        'version'           => 'latest'
    ];

    $rekognition = new RekognitionClient($options);

    // Get local image
    $photo = 'input.jpg';
    $fp_image = fopen($photo, 'r');
    $image = fread($fp_image, filesize($photo));
    fclose($fp_image);


    // Call DetectFaces
    $result = $rekognition->DetectFaces(array(
       'Image' => array(
          'Bytes' => $image,
       ),
       'Attributes' => array('ALL')
       )
```

```
    );

    // Display info for each detected person
    print 'People: Image position and estimated age' . PHP_EOL;
    for ($n=0;$n<sizeof($result['FaceDetails']); $n++){

      print 'Position: ' . $result['FaceDetails'][$n]['BoundingBox']['Left'] . "
 "
      . $result['FaceDetails'][$n]['BoundingBox']['Top']
      . PHP_EOL
      . 'Age (low): '.$result['FaceDetails'][$n]['AgeRange']['Low']
      .  PHP_EOL
      . 'Age (high): ' . $result['FaceDetails'][$n]['AgeRange']['High']
      .  PHP_EOL . PHP_EOL;
    }
?>
```

Ruby

Este ejemplo muestra una lista de las etiquetas que se han detectado en la imagen de entrada. Cambie el valor de `photo` por la ruta y el nombre de un archivo de imagen (en formato .jpg o .png).

```
#Copyright 2018 Amazon.com, Inc. or its affiliates. All Rights Reserved.
#PDX-License-Identifier: MIT-0 (For details, see https://github.com/awsdocs/
amazon-rekognition-developer-guide/blob/master/LICENSE-SAMPLECODE.)

    # gem 'aws-sdk-rekognition'
    require 'aws-sdk-rekognition'
    credentials = Aws::Credentials.new(
       ENV['AWS_ACCESS_KEY_ID'],
       ENV['AWS_SECRET_ACCESS_KEY']
    )
    client   = Aws::Rekognition::Client.new credentials: credentials
    photo = 'photo.jpg'
    path = File.expand_path(photo) # expand path relative to the current
 directory
    file = File.read(path)
    attrs = {
      image: {
        bytes: file
      },
      max_labels: 10
```

```
    }
    response = client.detect_labels attrs
    puts "Detected labels for: #{photo}"
    response.labels.each do |label|
      puts "Label:      #{label.name}"
      puts "Confidence: #{label.confidence}"
      puts "Instances:"
      label['instances'].each do |instance|
        box = instance['bounding_box']
        puts "  Bounding box:"
        puts "    Top:        #{box.top}"
        puts "    Left:       #{box.left}"
        puts "    Width:      #{box.width}"
        puts "    Height:     #{box.height}"
        puts "  Confidence: #{instance.confidence}"
      end
      puts "Parents:"
      label.parents.each do |parent|
        puts "  #{parent.name}"
      end
      puts "------------"
      puts ""
    end
```

Java V2

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

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.*;

import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
```

```
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DetectLabels {
    public static void main(String[] args) {
        final String usage = """
            Usage: <bucketName> <sourceImage>

            Where:
                bucketName - The name of the Amazon S3 bucket where the image is
 stored
                sourceImage - The name of the image file (for example,
 pic1.png).\s
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0] ;
        String sourceImage = args[1] ;
        Region region = Region.US_WEST_2;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        detectImageLabels(rekClient, bucketName, sourceImage);
        rekClient.close();
    }

    /**
     * Detects the labels in an image stored in an Amazon S3 bucket using the
 Amazon Rekognition service.
     *
     * @param rekClient     the Amazon Rekognition client used to make the
 detection request
     * @param bucketName    the name of the Amazon S3 bucket where the image is
 stored
     * @param sourceImage   the name of the image file to be analyzed
     */
```

```
    public static void detectImageLabels(RekognitionClient rekClient, String
 bucketName, String sourceImage) {
        try {
            S3Object s3ObjectTarget = S3Object.builder()
                    .bucket(bucketName)
                    .name(sourceImage)
                    .build();

            Image souImage = Image.builder()
                    .s3Object(s3ObjectTarget)
                    .build();

            DetectLabelsRequest detectLabelsRequest =
 DetectLabelsRequest.builder()
                    .image(souImage)
                    .maxLabels(10)
                    .build();

            DetectLabelsResponse labelsResponse =
 rekClient.detectLabels(detectLabelsRequest);
            List<Label> labels = labelsResponse.labels();
            System.out.println("Detected labels for the given photo");
            for (Label label : labels) {
                System.out.println(label.name() + ": " +
 label.confidence().toString());
            }

        } catch (RekognitionException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

Usando JavaScript

El siguiente ejemplo de JavaScript página web permite al usuario elegir una imagen y ver las edades estimadas de los rostros que se detectan en la imagen. Las edades estimadas se devuelven mediante una llamada a DetectFaces.

La imagen elegida se carga mediante la JavaScript `FileReader.readAsDataURL` función, que codifica la imagen en base64. Esto resulta útil para mostrar la imagen en un lienzo HTML.

Pero implica que los bytes de imagen se tienen que descodificar antes de transferirlos a una operación de Amazon Rekognition Image. En este ejemplo se muestra cómo descodificar los bytes de imagen subidos. Si los bytes de imagen codificados no le resultan útiles, use `FileReader.readAsArrayBuffer` en su lugar porque la imagen subida no está codificada. Esto significa que se puede llamar a las operaciones de Amazon Rekognition Image sin codificar primero los bytes de imagen. Para ver un ejemplo, consulta Uso de Buffer readAsArray.

Para ejecutar el ejemplo JavaScript

1. Cargue el código fuente de ejemplo en un editor.
2. Obtenga el identificador de grupo de identidades de Amazon Cognito. Para obtener más información, consulte Obtener el identificador de grupo de identidades de Amazon Cognito.
3. En la función `AnonLog` del código de ejemplo, cambie `IdentityPoolIdToUse` y `RegionToUse` por los valores que anotó en el paso 9 de Obtener el identificador de grupo de identidades de Amazon Cognito.
4. En la función `DetectFaces`, cambie `RegionToUse` por el valor que usó en el paso anterior.
5. Guarde el código fuente de ejemplo como un archivo `.html`.
6. Cargue el archivo en su navegador.
7. Haga clic en el botón Examinar... y elija una imagen que contenga una o más caras. Se muestra una tabla que contiene las edades estimadas para cada rostro detectado en la imagen.

> **Note**
>
> El siguiente ejemplo de código utiliza dos scripts que ya no forman parte de Amazon Cognito. Para obtener estos archivos, siga los enlaces para _aws-cognito-sdk.min.js y .min.js y amazon-cognito-identity, a continuación, guarde el texto de cada uno como archivos separados. `.js`

JavaScript código de ejemplo

El siguiente ejemplo de código usa JavaScript V2. Para ver un ejemplo de la JavaScript versión 3, consulte el ejemplo en el GitHub repositorio de ejemplos del SDK de AWS documentación.

```
<!--
Copyright 2018 Amazon.com, Inc. or its affiliates. All Rights Reserved.
```

```
PDX-License-Identifier: MIT-0 (For details, see https://github.com/awsdocs/amazon-
rekognition-developer-guide/blob/master/LICENSE-SAMPLECODE.)
-->
<!DOCTYPE html>
<html>
<head>
  <script src="aws-cognito-sdk.min.js"></script>
  <script src="amazon-cognito-identity.min.js"></script>
  <script src="https://sdk.amazonaws.com/js/aws-sdk-2.16.0.min.js"></script>
  <meta charset="UTF-8">
  <title>Rekognition</title>
</head>

<body>
  <H1>Age Estimator</H1>
  <input type="file" name="fileToUpload" id="fileToUpload" accept="image/*">
  <p id="opResult"></p>
</body>
<script>

  document.getElementById("fileToUpload").addEventListener("change", function (event) {
    ProcessImage();
  }, false);

  //Calls DetectFaces API and shows estimated ages of detected faces
  function DetectFaces(imageData) {
    AWS.region = "RegionToUse";
    var rekognition = new AWS.Rekognition();
    var params = {
      Image: {
        Bytes: imageData
      },
      Attributes: [
        'ALL',
      ]
    };
    rekognition.detectFaces(params, function (err, data) {
      if (err) console.log(err, err.stack); // an error occurred
      else {
       var table = "<table><tr><th>Low</th><th>High</th></tr>";
        // show each face and build out estimated age table
        for (var i = 0; i < data.FaceDetails.length; i++) {
          table += '<tr><td>' + data.FaceDetails[i].AgeRange.Low +
            '</td><td>' + data.FaceDetails[i].AgeRange.High + '</td></tr>';
```

```
        }
        table += "</table>";
        document.getElementById("opResult").innerHTML = table;
      }
    });
  }
  //Loads selected image and unencodes image bytes for Rekognition DetectFaces API
  function ProcessImage() {
    AnonLog();
    var control = document.getElementById("fileToUpload");
    var file = control.files[0];

    // Load base64 encoded image
    var reader = new FileReader();
    reader.onload = (function (theFile) {
      return function (e) {
        var img = document.createElement('img');
        var image = null;
        img.src = e.target.result;
        var jpg = true;
        try {
          image = atob(e.target.result.split("")[1]);

        } catch (e) {
          jpg = false;
        }
        if (jpg == false) {
          try {
            image = atob(e.target.result.split("")[1]);
          } catch (e) {
            alert("Not an image file Rekognition can process");
            return;
          }
        }
        //unencode image bytes for Rekognition DetectFaces API
        var length = image.length;
        imageBytes = new ArrayBuffer(length);
        var ua = new Uint8Array(imageBytes);
        for (var i = 0; i < length; i++) {
          ua[i] = image.charCodeAt(i);
        }
        //Call Rekognition
        DetectFaces(ua);
      };
```

```
    })(file);
    reader.readAsDataURL(file);
  }
  //Provides anonymous log on to AWS services
  function AnonLog() {

    // Configure the credentials provider to use your identity pool
    AWS.config.region = 'RegionToUse'; // Region
    AWS.config.credentials = new AWS.CognitoIdentityCredentials({
      IdentityPoolId: 'IdentityPoolIdToUse',
    });
    // Make the call to obtain credentials
    AWS.config.credentials.get(function () {
      // Credentials will be available when this function is called.
      var accessKeyId = AWS.config.credentials.accessKeyId;
      var secretAccessKey = AWS.config.credentials.secretAccessKey;
      var sessionToken = AWS.config.credentials.sessionToken;
    });
  }
</script>
</html>
```

Uso de Buffer readAsArray

El siguiente fragmento de código es una implementación alternativa de la `ProcessImage` función en el código de ejemplo, utilizando JavaScript V2. Utiliza `readAsArrayBuffer` para subir una imagen y llama a `DetectFaces`. Dado que `readAsArrayBuffer` no codifica en base64 el archivo subido, no es necesario descodificar los bytes de imagen antes de llamar a una operación de Amazon Rekognition Image.

```
//Copyright 2018 Amazon.com, Inc. or its affiliates. All Rights Reserved.
//PDX-License-Identifier: MIT-0 (For details, see https://github.com/awsdocs/amazon-
rekognition-developer-guide/blob/master/LICENSE-SAMPLECODE.)

function ProcessImage() {
    AnonLog();
    var control = document.getElementById("fileToUpload");
    var file = control.files[0];

    // Load base64 encoded image for display
    var reader = new FileReader();
    reader.onload = (function (theFile) {
      return function (e) {
```

```
        //Call Rekognition
        AWS.region = "RegionToUse";
        var rekognition = new AWS.Rekognition();
        var params = {
          Image: {
          Bytes: e.target.result
        },
        Attributes: [
        'ALL',
      ]
    };
    rekognition.detectFaces(params, function (err, data) {
      if (err) console.log(err, err.stack); // an error occurred
      else {
       var table = "<table><tr><th>Low</th><th>High</th></tr>";
        // show each face and build out estimated age table
        for (var i = 0; i < data.FaceDetails.length; i++) {
          table += '<tr><td>' + data.FaceDetails[i].AgeRange.Low +
            '</td><td>' + data.FaceDetails[i].AgeRange.High + '</td></tr>';
        }
        table += "</table>";
        document.getElementById("opResult").innerHTML = table;
      }
    });

      };
    })(file);
    reader.readAsArrayBuffer(file);
  }
```

Obtener el identificador de grupo de identidades de Amazon Cognito

Para simplificar, el ejemplo utiliza un grupo de identidades de Amazon Cognito anónimas para proporcionar acceso sin autenticar a la API de Amazon Rekognition Image. Esto podría ser adecuado para sus necesidades. Por ejemplo, puede utilizar el acceso sin autenticar para proporcionar acceso gratuito, o de prueba, a su sitio web antes de que los usuarios se inscriban. Para proporcionar acceso autenticado, utilice un grupo de usuarios de Amazon Cognito. Para obtener más información, consulte Grupos de usuarios de Amazon Cognito.

En el siguiente procedimiento se muestra cómo crear un grupo de identidades que permite el acceso a identidades sin autenticar y cómo obtener el identificador de grupo de identidades que se necesita en el código de ejemplo.

Para obtener el identificador de grupo de identidades

1. Abra la consola de Amazon Cognito.
2. Elija Crear nuevo grupo de identidades.
3. En Nombre de grupo de identidades*, escriba un nombre para su grupo de identidades.
4. En Identidades sin autenticar, elija Habilitar el acceso a identidades sin autenticar.
5. Elija Crear grupo.
6. Elija Ver detalles y anote el nombre de rol de las identidades sin autenticar.
7. Elija Permitir.
8. En Plataforma, elija. JavaScript
9. En Obtener credenciales de AWS, anote los valores de `AWS.config.region` y `IdentityPooldId` que se muestran en el fragmento de código.
10. Abra la consola de IAM en https://console.aws.amazon.com/iam/.
11. Seleccione Roles en el panel de navegación.
12. Elija el nombre de rol que anotó en el paso 6.
13. En la pestaña Permisos, elija Asociar políticas.
14. Elija AmazonRekognitionReadOnlyAccess.
15. Elija Adjuntar política.

Visualización de cuadros delimitadores

Las operaciones de Amazon Rekognition Image pueden devolver las coordenadas de los cuadros delimitadores de los elementos que se han detectado en las imágenes. Por ejemplo, la operación DetectFaces devuelve un cuadro delimitador (BoundingBox) para cada rostro detectado en una imagen. Puede utilizar las coordenadas del cuadro delimitador para mostrar un recuadro alrededor de los elementos detectados. Por ejemplo, en la imagen siguiente se muestra un cuadro delimitador que enmarca un rostro.



Un `BoundingBox` presenta las siguientes propiedades:

- Height: la altura del cuadro delimitador expresada proporcionalmente respecto a la altura total de la imagen.
- Left: la coordenada izquierda del cuadro delimitador expresada proporcionalmente respecto a la anchura total de la imagen.
- Top: la coordenada superior del cuadro delimitador expresada proporcionalmente respecto a la altura total de la imagen.

- Width: la anchura del cuadro delimitador expresada proporcionalmente respecto a la anchura total de la imagen.

Cada BoundingBox propiedad tiene un valor entre 0 y 1. El valor de cada propiedad es proporcional respecto a la anchura (`Left` y `Width`) o a la altura (`Height` y `Top`) totales de la imagen. Por ejemplo, si la imagen de entrada es de 700 x 200 píxeles y la coordenada superior izquierda del cuadro delimitador es de 350 x 50 píxeles, la API devuelve un valor de `Left` de 0,5 (350/700) y un valor de `Top` de 0,25 (50/200).

En el siguiente diagrama se muestra el rango de una imagen que cubre cada propiedad del cuadro delimitador.

Para mostrar el cuadro delimitador con la ubicación y el tamaño correctos, debe multiplicar BoundingBox los valores por el ancho o el alto de la imagen (según el valor que desee) para obtener los valores de los píxeles. Los valores en píxeles se utilizan para mostrar el cuadro delimitador. Por ejemplo, las dimensiones de la imagen anterior son 608 píxeles de anchura x 588 píxeles de altura. Los valores del cuadro delimitador del rostro son:

```
BoundingBox.Left: 0.3922065
BoundingBox.Top: 0.15567766
BoundingBox.Width: 0.284666
BoundingBox.Height: 0.2930403
```

La ubicación del cuadro delimitador del rostro en píxeles se calculan como se indica a continuación:

```
Left coordinate = BoundingBox.Left (0.3922065) * image width (608) = 238

Top coordinate = BoundingBox.Top (0.15567766) * image height (588) = 91

Face width = BoundingBox.Width (0.284666) * image width (608) = 173

Face height = BoundingBox.Height (0.2930403) * image height (588) = 172
```

Puede utilizar estos valores para mostrar un cuadro delimitador enmarcando el rostro.

> **Note**
>
> Una imagen se pueden orientar de varias formas. La aplicación podría tener que rotar la imagen para mostrarla con la orientación correcta. La orientación de la imagen afecta a las

coordenadas del cuadro delimitador. Es posible que tenga que traducir las coordenadas para poder mostrar el cuadro delimitador en la ubicación correcta. Para obtener más información, consulte Obtención de coordenadas de cuadro delimitador y orientación de imagen.

En los siguientes ejemplos se indica cómo mostrar un cuadro delimitador que enmarca los rostros detectados llamando a DetectFaces. En los ejemplos se da por hecho que las imágenes tienen una orientación de 0 grados. Además, en ellos también se indica cómo descargar la imagen desde un bucket de Amazon S3.

Para mostrar un cuadro delimitador

1. Si aún no lo ha hecho:
   a. Cree o actualice un usuario con los permisos `AmazonRekognitionFullAccess` y `AmazonS3ReadOnlyAccess`. Para obtener más información, consulte Paso 1: Configurar una cuenta de AWS y crear un usuario.
   b. Instale y configure el AWS CLI y el AWS SDKs. Para obtener más información, consulte Paso 2: Configure y AWS CLI AWS SDKs.
2. Consulte los siguientes ejemplos para llamar a la operación `DetectFaces`.

   Java

   Cambie el valor de `bucket` por el bucket de Amazon S3 que contiene el archivo de imagen. Cambie el valor de `photo` por el nombre de un archivo de imagen (en formato .jpg o .png).

```
//Loads images, detects faces and draws bounding boxes.Determines exif
 orientation, if necessary.
package com.amazonaws.samples;

//Import the basic graphics classes.
import java.awt.*;
import java.awt.image.BufferedImage;
import java.util.List;
import javax.imageio.ImageIO;
import javax.swing.*;

import com.amazonaws.services.rekognition.AmazonRekognition;
import com.amazonaws.services.rekognition.AmazonRekognitionClientBuilder;
```

```
import com.amazonaws.services.rekognition.model.BoundingBox;
import com.amazonaws.services.rekognition.model.DetectFacesRequest;
import com.amazonaws.services.rekognition.model.DetectFacesResult;
import com.amazonaws.services.rekognition.model.FaceDetail;
import com.amazonaws.services.rekognition.model.Image;
import com.amazonaws.services.rekognition.model.S3Object;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.S3ObjectInputStream;

// Calls DetectFaces and displays a bounding box around each detected image.
public class DisplayFaces extends JPanel {

    private static final long serialVersionUID = 1L;

    BufferedImage image;
    static int scale;
    DetectFacesResult result;

    public DisplayFaces(DetectFacesResult facesResult, BufferedImage bufImage)
 throws Exception {
        super();
        scale = 1; // increase to shrink image size.

        result = facesResult;
        image = bufImage;


    }
    // Draws the bounding box around the detected faces.
    public void paintComponent(Graphics g) {
        float left = 0;
        float top = 0;
        int height = image.getHeight(this);
        int width = image.getWidth(this);

        Graphics2D g2d = (Graphics2D) g; // Create a Java2D version of g.

        // Draw the image.
        g2d.drawImage(image, 0, 0, width / scale, height / scale, this);
        g2d.setColor(new Color(0, 212, 0));

        // Iterate through faces and display bounding boxes.
```

```
        List<FaceDetail> faceDetails = result.getFaceDetails();
        for (FaceDetail face : faceDetails) {

            BoundingBox box = face.getBoundingBox();
            left = width * box.getLeft();
            top = height * box.getTop();
            g2d.drawRect(Math.round(left / scale), Math.round(top / scale),
                    Math.round((width * box.getWidth()) / scale),
Math.round((height * box.getHeight())) / scale);

        }
    }


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

        String photo = "photo.png";
        String bucket = "bucket";
        int height = 0;
        int width = 0;

        // Get the image from an S3 Bucket
        AmazonS3 s3client = AmazonS3ClientBuilder.defaultClient();

        com.amazonaws.services.s3.model.S3Object s3object =
s3client.getObject(bucket, photo);
        S3ObjectInputStream inputStream = s3object.getObjectContent();
        BufferedImage image = ImageIO.read(inputStream);
        DetectFacesRequest request = new DetectFacesRequest()
                .withImage(new Image().withS3Object(new
S3Object().withName(photo).withBucket(bucket)));

        width = image.getWidth();
        height = image.getHeight();

        // Call DetectFaces
        AmazonRekognition amazonRekognition =
AmazonRekognitionClientBuilder.defaultClient();
        DetectFacesResult result = amazonRekognition.detectFaces(request);

        //Show the bounding box info for each face.
        List<FaceDetail> faceDetails = result.getFaceDetails();
        for (FaceDetail face : faceDetails) {
```

```
            BoundingBox box = face.getBoundingBox();
            float left = width * box.getLeft();
            float top = height * box.getTop();
            System.out.println("Face:");

            System.out.println("Left: " + String.valueOf((int) left));
            System.out.println("Top: " + String.valueOf((int) top));
            System.out.println("Face Width: " + String.valueOf((int) (width *
 box.getWidth())));
            System.out.println("Face Height: " + String.valueOf((int) (height *
 box.getHeight())));
            System.out.println();

        }

        // Create frame and panel.
        JFrame frame = new JFrame("RotateImage");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        DisplayFaces panel = new DisplayFaces(result, image);
        panel.setPreferredSize(new Dimension(image.getWidth() / scale,
 image.getHeight() / scale));
        frame.setContentPane(panel);
        frame.pack();
        frame.setVisible(true);

    }
}
```

Python

Cambie el valor de `bucket` por el bucket de Amazon S3 que contiene el archivo de imagen. Cambie el valor de `photo` por el nombre de un archivo de imagen (en formato .jpg o .png). Sustituya el valor de `profile_name` en la línea que crea la sesión de Rekognition por el nombre de su perfil de desarrollador.

```
import boto3
import io
from PIL import Image, ImageDraw

def show_faces(photo, bucket):

    session = boto3.Session(profile_name='profile-name')
```

```
    client = session.client('rekognition')

    # Load image from S3 bucket
    s3_connection = boto3.resource('s3')
    s3_object = s3_connection.Object(bucket, photo)
    s3_response = s3_object.get()

    stream = io.BytesIO(s3_response['Body'].read())
    image = Image.open(stream)

    # Call DetectFaces
    response = client.detect_faces(Image={'S3Object': {'Bucket': bucket, 'Name':
 photo}},
                                   Attributes=['ALL'])

    imgWidth, imgHeight = image.size
    draw = ImageDraw.Draw(image)

    # calculate and display bounding boxes for each detected face
    print('Detected faces for ' + photo)
    for faceDetail in response['FaceDetails']:
        print('The detected face is between ' + str(faceDetail['AgeRange']
['Low'])
              + ' and ' + str(faceDetail['AgeRange']['High']) + ' years old')

        box = faceDetail['BoundingBox']
        left = imgWidth * box['Left']
        top = imgHeight * box['Top']
        width = imgWidth * box['Width']
        height = imgHeight * box['Height']

        print('Left: ' + '{0:.0f}'.format(left))
        print('Top: ' + '{0:.0f}'.format(top))
        print('Face Width: ' + "{0:.0f}".format(width))
        print('Face Height: ' + "{0:.0f}".format(height))

        points = (
            (left, top),
            (left + width, top),
            (left + width, top + height),
            (left, top + height),
            (left, top)

        )
```

```
        draw.line(points, fill='#00d400', width=2)

        # Alternatively can draw rectangle. However you can't set line width.
        # draw.rectangle([left,top, left + width, top + height],
 outline='#00d400')

    image.show()

    return len(response['FaceDetails'])

def main():
    bucket = "bucket-name"
    photo = "photo-name"
    faces_count = show_faces(photo, bucket)
    print("faces detected: " + str(faces_count))

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.

Tenga en cuenta que `s3` se refiere al cliente de Amazon S3 del SDK de AWS y `rekClient` al cliente de Amazon Rekognition del SDK de AWS.

```
//snippet-start:[rekognition.java2.detect_labels.import]
import java.awt.*;
import java.awt.image.BufferedImage;
import java.io.ByteArrayInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.io.InputStream;
import java.util.List;
import javax.imageio.ImageIO;
import javax.swing.*;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.core.ResponseBytes;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.model.Attribute;
import software.amazon.awssdk.services.rekognition.model.BoundingBox;
```

```
import software.amazon.awssdk.services.rekognition.model.DetectFacesRequest;
import software.amazon.awssdk.services.rekognition.model.DetectFacesResponse;
import software.amazon.awssdk.services.rekognition.model.FaceDetail;
import software.amazon.awssdk.services.rekognition.model.Image;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.GetObjectResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;
//snippet-end:[rekognition.java2.detect_labels.import]

/**
* Before running this Java V2 code example, set up your development environment,
 including your credentials.
*
* For more information, see the following documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
*/
public class DisplayFaces extends JPanel {

    static DetectFacesResponse result;
    static BufferedImage image;
    static int scale;

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

        final String usage = "\n" +
            "Usage: " +
            "   <sourceImage> <bucketName>\n\n" +
            "Where:\n" +
            "   sourceImage - The name of the image in an Amazon S3 bucket (for
 example, people.png). \n\n" +
            "   bucketName - The name of the Amazon S3 bucket (for example,
 amzn-s3-demo-bucket). \n\n";

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String sourceImage = args[0];
```

```
        String bucketName = args[1];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .credentialsProvider(ProfileCredentialsProvider.create("profile-
name"))
            .build();

        RekognitionClient rekClient = RekognitionClient.builder()
            .region(region)
            .credentialsProvider(ProfileCredentialsProvider.create("profile-
name"))
            .build();

        displayAllFaces(s3, rekClient, sourceImage, bucketName);
        s3.close();
        rekClient.close();
    }

    // snippet-start:[rekognition.java2.display_faces.main]
    public static void displayAllFaces(S3Client s3,
                                       RekognitionClient rekClient,
                                       String sourceImage,
                                       String bucketName) {
        int height;
        int width;
        byte[] data = getObjectBytes (s3, bucketName, sourceImage);
        InputStream is = new ByteArrayInputStream(data);

        try {
            SdkBytes sourceBytes = SdkBytes.fromInputStream(is);
            image = ImageIO.read(sourceBytes.asInputStream());
            width = image.getWidth();
            height = image.getHeight();

            // Create an Image object for the source image
            software.amazon.awssdk.services.rekognition.model.Image souImage =
 Image.builder()
                .bytes(sourceBytes)
                .build();

            DetectFacesRequest facesRequest = DetectFacesRequest.builder()
                .attributes(Attribute.ALL)
                .image(souImage)
```

```
                .build();

            result = rekClient.detectFaces(facesRequest);

            // Show the bounding box info for each face.
            List<FaceDetail> faceDetails = result.faceDetails();
            for (FaceDetail face : faceDetails) {
                BoundingBox box = face.boundingBox();
                float left = width * box.left();
                float top = height * box.top();
                System.out.println("Face:");

                System.out.println("Left: " + (int) left);
                System.out.println("Top: " + (int) top);
                System.out.println("Face Width: " + (int) (width *
box.width()));
                System.out.println("Face Height: " + (int) (height *
box.height()));
                System.out.println();
            }

            // Create the frame and panel.
            JFrame frame = new JFrame("RotateImage");
            frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
            DisplayFaces panel = new DisplayFaces(image);
            panel.setPreferredSize(new Dimension(image.getWidth() / scale,
image.getHeight() / scale));
            frame.setContentPane(panel);
            frame.pack();
            frame.setVisible(true);

        } catch (RekognitionException | FileNotFoundException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        } catch (IOException e) {
           e.printStackTrace();
        }
    }

    public static byte[] getObjectBytes (S3Client s3, String bucketName, String
keyName) {

        try {
            GetObjectRequest objectRequest = GetObjectRequest
```

```
                .builder()
                .key(keyName)
                .bucket(bucketName)
                .build();

            ResponseBytes<GetObjectResponse> objectBytes =
s3.getObjectAsBytes(objectRequest);
            return objectBytes.asByteArray();

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return null;
    }

    public DisplayFaces(BufferedImage bufImage) {
        super();
        scale = 1; // increase to shrink image size.
        image = bufImage;
    }

    // Draws the bounding box around the detected faces.
    public void paintComponent(Graphics g) {
        float left;
        float top;
        int height = image.getHeight(this);
        int width = image.getWidth(this);
        Graphics2D g2d = (Graphics2D) g; // Create a Java2D version of g.

        // Draw the image
        g2d.drawImage(image, 0, 0, width / scale, height / scale, this);
        g2d.setColor(new Color(0, 212, 0));

        // Iterate through the faces and display bounding boxes.
        List<FaceDetail> faceDetails = result.faceDetails();
        for (FaceDetail face : faceDetails) {
            BoundingBox box = face.boundingBox();
            left = width * box.left();
            top = height * box.top();
            g2d.drawRect(Math.round(left / scale), Math.round(top / scale),
                    Math.round((width * box.width()) / scale),
Math.round((height * box.height())) / scale);
        }
```

```
    }
    // snippet-end:[rekognition.java2.display_faces.main]
}
```

Obtención de coordenadas de cuadro delimitador y orientación de imagen

Las aplicaciones que utilizan Amazon Rekognition Image habitualmente tienen que mostrar las imágenes detectadas por las operaciones de Amazon Rekognition Image y los cuadros alrededor de rostros detectados. Para visualizar una imagen correctamente en la aplicación, tiene que conocer la orientación. Tal vez tenga que corregir esta orientación. Para algunos archivos .jpg, la orientación de la imagen está contenida en los metadatos de formato de archivo de imagen intercambiable (EXIF) de la imagen.

Para mostrar un cuadro alrededor de un rostro, necesita las coordenadas del cuadro delimitador del rostro. Si el cuadro no está orientado correctamente, es posible que tenga que ajustar esas coordenadas. Las operaciones de detección de rostros de Amazon Rekognition Image devuelven las coordenadas del cuadro delimitador de cada rostro detectado, pero no estiman las coordenadas de los archivos.jpg sin metadatos Exif.

Los siguientes ejemplos muestran cómo obtener las coordenadas del cuadro delimitador para los rostros detectados en una imagen.

Utilice la información de este ejemplo para garantizar que sus imágenes están correctamente orientadas y que los cuadros delimitadores se muestran en la ubicación correcta en su aplicación.

Dado que el código que se utiliza para girar y mostrar imágenes y cuadros delimitadores depende del lenguaje y del entorno que utilice, no se explica cómo mostrar imágenes y cuadros delimitadores en el código o cómo obtener la información de orientación a partir de los metadatos Exif.

Búsqueda de la orientación de una imagen

Para mostrar una imagen correctamente en su aplicación, es posible que tenga que girarla. La siguiente imagen está orientada a 0 grados y se muestra correctamente.



Sin embargo, la siguiente imagen está girada 90 grados en sentido contrario a las agujas del reloj. Para mostrarla correctamente, tiene que encontrar la orientación de la imagen y utilizar dicha información en su código para girar la imagen a 0 grados.



Algunas imágenes en formato .jpg contienen información de orientación en metadatos Exif. Si están disponibles, los metadatos Exif de la imagen contienen la orientación. En los metadatos Exif, puede encontrar la orientación de la imagen en el campo `orientation`. Aunque Amazon Rekognition Image identifica la presencia de información de orientación de imagen en los metadatos Exif, no proporciona acceso a la misma. Para acceder a los metadatos Exif en una imagen, utilice una biblioteca de terceros o escriba su propio código. Para obtener más información, consulte Exif Version 2.32.

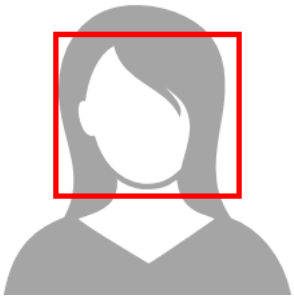
Cuando se conoce la orientación de una imagen, puede escribir código para girarla y mostrarla correctamente.

Visualización de cuadros delimitadores

Las operaciones de Amazon Rekognition Image que analizan rostros en una imagen devuelven además las coordenadas de los cuadros delimitadores que rodean los rostros. Para obtener más información, consulte BoundingBox.

Para mostrar un cuadro delimitador alrededor de un rostro similar al cuadro mostrado en la imagen siguiente en su aplicación, utilice las coordenadas de cuadro delimitador en su código. Las

coordenadas del cuadro delimitador devueltas por una operación reflejan la orientación de la imagen. Si tiene que girar la imagen para mostrarla correctamente, es posible que tenga que traducir las coordenadas del cuadro delimitador.



Mostrar cuadros delimitadores cuando la información de orientación está presente en los metadatos Exif

Si la orientación de una imagen está incluida en metadatos Exif, las operaciones de Amazon Rekognition Image hacen lo siguiente:

- Devolver nulo en el campo de corrección de orientación en la respuesta de la operación. Para girar la imagen, utilice la orientación proporcionada en los metadatos Exif en el código.
- Devuelva las coordenadas del cuadro delimitador ya orientadas a 0 grados. Para mostrar el cuadro delimitador en la posición correcta, utilice las coordenadas que se devolvieron. No tiene que traducirlas.

Ejemplo: obtención de coordenadas de cuadro delimitador y orientación de imagen para una imagen

En los siguientes ejemplos se muestra cómo se utiliza el SDK de AWS con el fin de obtener los datos de orientación de una imagen Exif y las coordenadas de los cuadros delimitadores para los famosos detectados mediante la operación `RecognizeCelebrities`.

> **Note**
>
> El soporte para estimar la orientación de la imagen mediante el campo `OrientationCorrection` cesó en agosto de 2021. Todos los valores devueltos para este campo incluidos en una respuesta de la API siempre serán NULL.

Java

Este ejemplo carga una imagen del sistema de archivos local, llama a la operación `RecognizeCelebrities`, determina la altura y la anchura de la imagen y calcula las coordenadas del cuadro delimitador del rostro para la imagen girada. El ejemplo no muestra cómo procesar la información de orientación que hay almacenada en los metadatos Exif.

En la función `main`, cambie el valor de `photo` por el nombre y la ruta de una imagen que esté almacenada localmente en formato .png o .jpg.

```
//Copyright 2018 Amazon.com, Inc. or its affiliates. All Rights Reserved.
//PDX-License-Identifier: MIT-0 (For details, see https://github.com/awsdocs/amazon-rekognition-developer-guide/blob/master/LICENSE-SAMPLECODE.)

package com.amazonaws.samples;
import java.awt.image.BufferedImage;
import java.io.ByteArrayInputStream;
import java.io.ByteArrayOutputStream;
import java.io.File;
import java.io.FileInputStream;
import java.io.InputStream;
import java.nio.ByteBuffer;
import java.util.List;
import javax.imageio.ImageIO;
import com.amazonaws.services.rekognition.AmazonRekognition;
import com.amazonaws.services.rekognition.AmazonRekognitionClientBuilder;
import com.amazonaws.services.rekognition.model.Image;
import com.amazonaws.services.rekognition.model.RecognizeCelebritiesRequest;
import com.amazonaws.services.rekognition.model.RecognizeCelebritiesResult;
import com.amazonaws.util.IOUtils;
import com.amazonaws.services.rekognition.model.AmazonRekognitionException;
import com.amazonaws.services.rekognition.model.BoundingBox;
import com.amazonaws.services.rekognition.model.Celebrity;
import com.amazonaws.services.rekognition.model.ComparedFace;

public class RotateImage {

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

  String photo = "photo.png";

  //Get Rekognition client
```

```
AmazonRekognition amazonRekognition =
AmazonRekognitionClientBuilder.defaultClient();


 // Load image
 ByteBuffer imageBytes=null;
 BufferedImage image = null;

 try (InputStream inputStream = new FileInputStream(new File(photo))) {
    imageBytes = ByteBuffer.wrap(IOUtils.toByteArray(inputStream));

 }
 catch(Exception e)
 {
     System.out.println("Failed to load file " + photo);
     System.exit(1);
 }

 //Get image width and height
 InputStream imageBytesStream;
 imageBytesStream = new ByteArrayInputStream(imageBytes.array());

 ByteArrayOutputStream baos = new ByteArrayOutputStream();
 image=ImageIO.read(imageBytesStream);
 ImageIO.write(image, "jpg", baos);

 int height = image.getHeight();
 int width = image.getWidth();

 System.out.println("Image Information:");
 System.out.println(photo);
 System.out.println("Image Height: " + Integer.toString(height));
 System.out.println("Image Width: " + Integer.toString(width));

 //Call GetCelebrities

 try{
   RecognizeCelebritiesRequest request = new RecognizeCelebritiesRequest()
          .withImage(new Image()
             .withBytes((imageBytes)));


     RecognizeCelebritiesResult result =
amazonRekognition.recognizeCelebrities(request);
```

```
      // The returned value of OrientationCorrection will always be null
      System.out.println("Orientation: " + result.getOrientationCorrection() +
 "\n");
      List <Celebrity> celebs = result.getCelebrityFaces();

      for (Celebrity celebrity: celebs) {
          System.out.println("Celebrity recognized: " + celebrity.getName());
          System.out.println("Celebrity ID: " + celebrity.getId());
          ComparedFace  face = celebrity.getFace()
;             ShowBoundingBoxPositions(height,
                  width,
                  face.getBoundingBox(),
                  result.getOrientationCorrection());

            System.out.println();
       }

   } catch (AmazonRekognitionException e) {
      e.printStackTrace();
   }

}


public static void ShowBoundingBoxPositions(int imageHeight, int imageWidth,
 BoundingBox box, String rotation) {

  float left = 0;
  float top = 0;

  if(rotation==null){
      System.out.println("No estimated estimated orientation. Check Exif data.");
      return;
  }
  //Calculate face position based on image orientation.
  switch (rotation) {
     case "ROTATE_0":
        left = imageWidth * box.getLeft();
        top = imageHeight * box.getTop();
        break;
     case "ROTATE_90":
        left = imageHeight * (1 - (box.getTop() + box.getHeight()));
        top = imageWidth * box.getLeft();
        break;
```

```
     case "ROTATE_180":
        left = imageWidth - (imageWidth * (box.getLeft() + box.getWidth()));
        top = imageHeight * (1 - (box.getTop() + box.getHeight()));
        break;
     case "ROTATE_270":
        left = imageHeight * box.getTop();
        top = imageWidth * (1 - box.getLeft() - box.getWidth());
        break;
     default:
        System.out.println("No estimated orientation information. Check Exif
 data.");
        return;
  }

  //Display face location information.
  System.out.println("Left: " + String.valueOf((int) left));
  System.out.println("Top: " + String.valueOf((int) top));
  System.out.println("Face Width: " + String.valueOf((int)(imageWidth *
 box.getWidth())));
  System.out.println("Face Height: " + String.valueOf((int)(imageHeight *
 box.getHeight())));

  }
}
```

Python

En este ejemplo, se usa la biblioteca de PIL/Pillow imágenes para obtener el ancho y el alto de la imagen. Para obtener más información, consulte Pillow. En este ejemplo se conservan los metadatos exif porque los puede necesitar en otra parte de la aplicación.

En la función `main`, cambie el valor de `photo` por el nombre y la ruta de una imagen que esté almacenada localmente en formato .png o .jpg.

```
#Copyright 2018 Amazon.com, Inc. or its affiliates. All Rights Reserved.
#PDX-License-Identifier: MIT-0 (For details, see https://github.com/awsdocs/amazon-
rekognition-developer-guide/blob/master/LICENSE-SAMPLECODE.)

import boto3
import io
from PIL import Image
```

```
# Calculate positions from from estimated rotation
def show_bounding_box_positions(imageHeight, imageWidth, box):
    left = 0
    top = 0

    print('Left: ' + '{0:.0f}'.format(left))
    print('Top: ' + '{0:.0f}'.format(top))
    print('Face Width: ' + "{0:.0f}".format(imageWidth * box['Width']))
    print('Face Height: ' + "{0:.0f}".format(imageHeight * box['Height']))


def celebrity_image_information(photo):
    client = boto3.client('rekognition')

    # Get image width and height
    image = Image.open(open(photo, 'rb'))
    width, height = image.size

    print('Image information: ')
    print(photo)
    print('Image Height: ' + str(height))
    print('Image Width: ' + str(width))

    # call detect faces and show face age and placement
    # if found, preserve exif info
    stream = io.BytesIO()
    if 'exif' in image.info:
        exif = image.info['exif']
        image.save(stream, format=image.format, exif=exif)
    else:
        image.save(stream, format=image.format)
    image_binary = stream.getvalue()

    response = client.recognize_celebrities(Image={'Bytes': image_binary})

    print()
    print('Detected celebrities for ' + photo)

    for celebrity in response['CelebrityFaces']:
        print('Name: ' + celebrity['Name'])
        print('Id: ' + celebrity['Id'])

        # Value of "orientation correction" will always be null
```

```
        if 'OrientationCorrection' in response:
            show_bounding_box_positions(height, width, celebrity['Face']
['BoundingBox'])

        print()
    return len(response['CelebrityFaces'])


def main():
    photo = 'photo'

    celebrity_count = celebrity_image_information(photo)
    print("celebrities detected: " + str(celebrity_count))


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.

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import
 software.amazon.awssdk.services.rekognition.model.RecognizeCelebritiesRequest;
import software.amazon.awssdk.services.rekognition.model.Image;
import
 software.amazon.awssdk.services.rekognition.model.RecognizeCelebritiesResponse;
import software.amazon.awssdk.services.rekognition.model.Celebrity;
import software.amazon.awssdk.services.rekognition.model.ComparedFace;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import software.amazon.awssdk.services.rekognition.model.BoundingBox;
import javax.imageio.ImageIO;
import java.awt.image.BufferedImage;
import java.io.*;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
```

```
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class RotateImage {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <sourceImage>

                Where:
                   sourceImage - The path to the image (for example, C:\\AWS\
\pic1.png).\s
                """;

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

        String sourceImage = args[0];
        Region region = Region.US_WEST_2;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        System.out.println("Locating celebrities in " + sourceImage);
        recognizeAllCelebrities(rekClient, sourceImage);
        rekClient.close();
    }

    public static void recognizeAllCelebrities(RekognitionClient rekClient, String
 sourceImage) {
        try {
            BufferedImage image;
            InputStream sourceStream = new FileInputStream(sourceImage);
            SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);

            image = ImageIO.read(sourceBytes.asInputStream());
            int height = image.getHeight();
            int width = image.getWidth();

            Image souImage = Image.builder()
```

```
                    .bytes(sourceBytes)
                    .build();

            RecognizeCelebritiesRequest request =
RecognizeCelebritiesRequest.builder()
                    .image(souImage)
                    .build();

            RecognizeCelebritiesResponse result =
rekClient.recognizeCelebrities(request);
            List<Celebrity> celebs = result.celebrityFaces();
            System.out.println(celebs.size() + " celebrity(s) were recognized.\n");
            for (Celebrity celebrity : celebs) {
                System.out.println("Celebrity recognized: " + celebrity.name());
                System.out.println("Celebrity ID: " + celebrity.id());
                ComparedFace face = celebrity.face();
                ShowBoundingBoxPositions(height,
                        width,
                        face.boundingBox(),
                        result.orientationCorrectionAsString());
            }

        } catch (RekognitionException | FileNotFoundException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    public static void ShowBoundingBoxPositions(int imageHeight, int imageWidth,
BoundingBox box, String rotation) {
        float left;
        float top;
        if (rotation == null) {
            System.out.println("No estimated estimated orientation.");
            return;
        }

        // Calculate face position based on the image orientation.
        switch (rotation) {
            case "ROTATE_0" -> {
                left = imageWidth * box.left();
                top = imageHeight * box.top();
```

```
            }
            case "ROTATE_90" -> {
                left = imageHeight * (1 - (box.top() + box.height()));
                top = imageWidth * box.left();
            }
            case "ROTATE_180" -> {
                left = imageWidth - (imageWidth * (box.left() + box.width()));
                top = imageHeight * (1 - (box.top() + box.height()));
            }
            case "ROTATE_270" -> {
                left = imageHeight * box.top();
                top = imageWidth * (1 - box.left() - box.width());
            }
            default -> {
                System.out.println("No estimated orientation information. Check Exif
 data.");
                return;
            }
        }

        System.out.println("Left: " + (int) left);
        System.out.println("Top: " + (int) top);
        System.out.println("Face Width: " + (int) (imageWidth * box.width()));
        System.out.println("Face Height: " + (int) (imageHeight * box.height()));
    }
}
```

Cómo trabajar con operaciones de análisis de vídeos almacenados

Amazon Rekognition Video es un API que puede utilizar para analizar vídeos. Con Amazon Rekognition Video, puede detectar etiquetas, rostros, personas, famosos y contenido para adultos (insinuante y explícito) en los vídeos que se almacenan en un bucket de Amazon Simple Storage Service (Amazon S3). Puede utilizar Amazon Rekognition Video en categorías como seguridad pública. media/entertainment Anteriormente, la exploración de vídeos para detectar objetos o personas habría conllevado muchas horas de visualización por parte de un ser humano proclive a errores. Amazon Rekognition Video automatiza la detección de elementos y cuándo se producen a lo largo de un vídeo.

En esta sección, se describen los tipos de análisis que Amazon Rekognition Video puede realizar, se incluye información general sobre la API y se muestran ejemplos de uso de Amazon Rekognition Video.

Temas

- Tipos de análisis
- Descripción general de la API de Amazon Rekognition Video
- Cómo llamar a las operaciones de Amazon Rekognition Video
- Configuración de Amazon Rekognition Video
- Análisis de un vídeo almacenado en un bucket de Amazon S3 con Java o Python (SDK)
- Analizar un vídeo con el AWS Command Line Interface
- Referencia: notificación de resultados de análisis de vídeo
- Solución de problemas de Amazon Rekognition Video

Tipos de análisis

Puede utilizar Amazon Rekognition Video para analizar vídeos para obtener la siguiente información:

- Segmentos de vídeo
- Etiquetas
- Contenido para adultos insinuante y explícito
- Text
- Famosos
- Rostros
- Personas

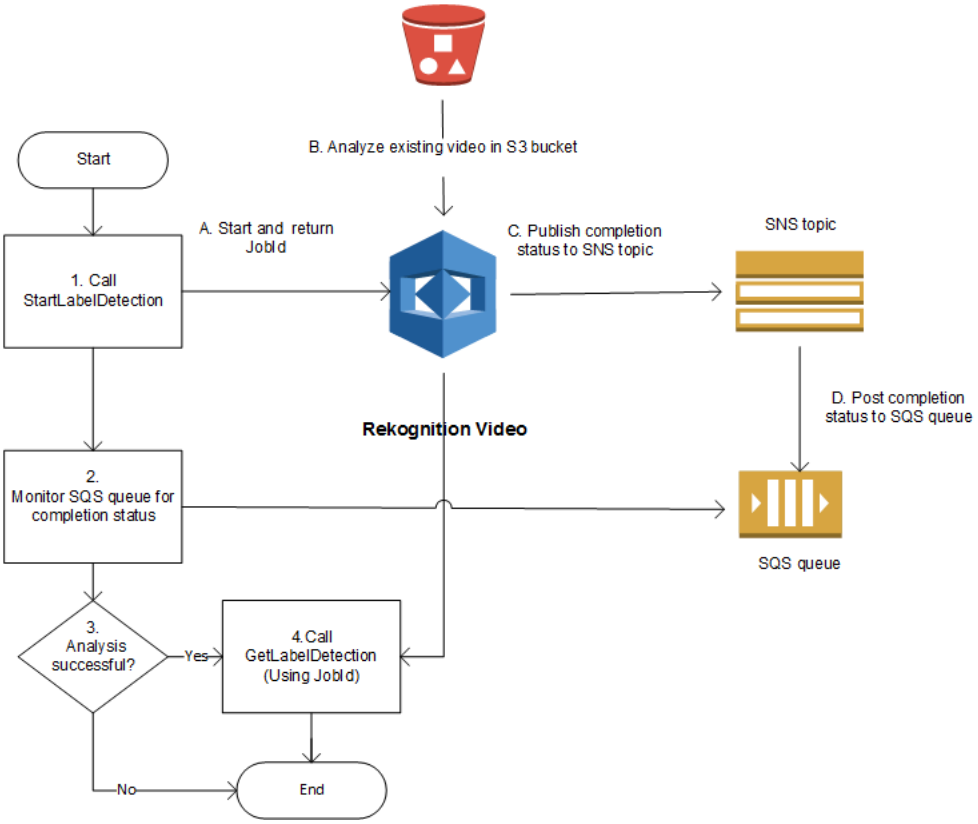
Para obtener más información, consulte Cómo funciona Amazon Rekognition.

Descripción general de la API de Amazon Rekognition Video

Amazon Rekognition Video procesa un vídeo que está almacenado en un bucket de Amazon S3. El patrón de diseño es un conjunto asíncrono de operaciones. El análisis de vídeo comienza llamando a una operación `Start` como, por ejemplo, StartLabelDetection. El estado de finalización

de la solicitud se publica en un tema de Amazon Simple Notification Service (Amazon SNS). Para obtener el estado de finalización del tema Amazon SNS, puede utilizar una cola del Amazon Simple Queue Service (Amazon SQS) o una función. AWS Lambda Una vez que disponga del estado de realización, llame a una operación `Get`, como GetLabelDetection, para obtener los resultados de la solicitud.

En el siguiente diagrama se muestra el proceso para detectar etiquetas en un vídeo que está almacenado en un bucket de Amazon S3. En el diagrama, una cola de Amazon SQS obtiene el estado de realización a partir del tema de Amazon SNS. Como alternativa, puede utilizar una función. AWS Lambda



El proceso es el mismo para otras operaciones de Amazon Rekognition Video. La siguiente tabla muestra las operaciones `Start` y `Get` para cada una de las operaciones Amazon Rekognition sin almacenamiento.

Detección	Operación START	Operación GET
Segmentos de vídeo	StartSegmentDetection	GetSegmentDetection
Etiquetas	StartLabelDetection	GetLabelDetection

Detección	Operación START	Operación GET
Contenido para adultos explícito o insinuante	StartContentModeration	GetContentModeration
Texto	StartTextDetection	GetTextDetection
Famosos	StartCelebrityRecognition	GetCelebrityRecognition
Rostros	StartFaceDetection	GetFaceDetection
People	StartPersonTracking	GetPersonTracking

Para operaciones `Get` distintas de `GetCelebrityRecognition`, Amazon Rekognition Video devuelve información de seguimiento para cuando se detectan entidades a lo largo del vídeo de entrada.

Para obtener más información sobre el uso de Amazon Rekognition Video, consulte Cómo llamar a las operaciones de Amazon Rekognition Video. Para ver un ejemplo que realiza análisis de vídeo mediante la utilización de Amazon SQS, consulte Análisis de un vídeo almacenado en un bucket de Amazon S3 con Java o Python (SDK). Analizar un vídeo con el AWS Command Line InterfacePara ver algunos ejemplos, consulte. AWS CLI

Formatos de vídeo y almacenamiento

Las operaciones de Amazon Rekognition pueden analizar vídeos que están almacenados en buckets de Amazon S3. Para obtener una lista de todos los límites de la operación de análisis de vídeo, consulte Directrices y cuotas.

El vídeo se debe codificar utilizando el códec H.264. Los formatos de archivo admitidos son MPEG-4 y MOV.

Un códec es un software o hardware que comprime datos para una entrega más rápida y descomprime los datos recibidos a su forma original. El códec H.264 suele utilizarse habitualmente para grabar, comprimir y distribuir contenido de vídeo. Un formato de archivo de vídeo puede contener uno o varios códecs. Si su archivo de vídeo de formato MOV o MPEG-4 no funciona con Amazon Rekognition Video, compruebe que el códec utilizado para cifrar el vídeo sea H.264.

Cualquier API de Amazon Rekognition Video que analice datos de audio solo admite códecs de audio AAC.

El tamaño de archivo máximo para un vídeo almacenado es de 10 GB.

Búsqueda de personas

Puede utilizar metadatos faciales que están almacenados en una colección para buscar personas en un vídeo. Por ejemplo, puede buscar en un vídeo archivado a una persona concreta o a varias personas. Puede almacenar los metadatos faciales desde imágenes de origen en una colección utilizando la operación IndexFaces. A continuación, puede utilizar StartFaceSearch para comenzar a buscar asíncronamente rostros en la colección. Puede utilizar GetFaceSearch para obtener los resultados de la búsqueda. Para obtener más información, consulte Búsqueda de rostros en vídeos almacenados. La búsqueda de personas es un ejemplo de operación de Amazon Rekognition basada en almacenamiento. Para obtener más información, consulte Operaciones de API con almacenamiento.

También puede buscar personas en un vídeo en streaming. Para obtener más información, consulte Trabajar con eventos de vídeo en streaming.

Cómo llamar a las operaciones de Amazon Rekognition Video

Amazon Rekognition Video es una API asincrónica que puede utilizar para analizar vídeos almacenados en un bucket de Amazon Simple Storage Service (Amazon S3). Para iniciar el análisis de un vídeo, llame a una operación de Amazon Rekognition `Start` Video, como. StartPersonTracking Amazon Rekognition Video publica el resultado de la solicitud de análisis en un tema de Amazon Simple Notification Service (Amazon SNS). Puede utilizar una cola del Amazon Simple Queue Service (Amazon SQS) o AWS Lambda una función para obtener el estado de finalización de la solicitud de análisis de vídeo del tema Amazon SNS. Por último, puede obtener los resultados de la solicitud de análisis de vídeo llamando a una operación de Amazon Rekognition`Get`, como. GetPersonTracking

La información en las secciones siguientes utiliza las operaciones de detección de etiquetas para mostrar cómo detecta Amazon Rekognition Video etiquetas (objetos, eventos, conceptos y actividades) en un vídeo que está almacenado en un bucket de Amazon S3. El mismo enfoque funciona para las demás operaciones de Amazon Rekognition Video, por ejemplo, y. StartFaceDetectionStartPersonTracking El ejemplo Análisis de un vídeo almacenado en un bucket de Amazon S3 con Java o Python (SDK) muestra cómo analizar un vídeo mediante el uso de una cola de Amazon SQS para obtener el estado de la realización a partir del tema de Amazon SNS. También se utiliza como base para otros ejemplos de Amazon Rekognition Video, como Recorridos de las personas. Para ver ejemplos, consulte. AWS CLI Analizar un vídeo con el AWS Command Line Interface

Temas

- Comenzar el análisis de vídeo
- Obtención del estado de realización de una solicitud de análisis de Amazon Rekognition Video
- Obtención de los resultados del análisis de Amazon Rekognition Video

Comenzar el análisis de vídeo

Para iniciar una solicitud de detección de etiquetas de Amazon Rekognition Video, debe llamar. StartLabelDetection El siguiente es un ejemplo de una solicitud JSON que ha transferido `StartLabelDetection`.

```
{
    "Video": {
        "S3Object": {
            "Bucket": "amzn-s3-demo-bucket",
            "Name": "video.mp4"
        }
    },
    "ClientRequestToken": "LabelDetectionToken",
    "MinConfidence": 50,
    "NotificationChannel": {
        "SNSTopicArn": "arn:aws:sns:us-east-1:nnnnnnnnnn:topic",
        "RoleArn": "arn:aws:iam::nnnnnnnnnn:role/roleopic"
    },
    "JobTag": "DetectingLabels"
}
```

El parámetro de entrada `Video` proporciona el nombre del archivo de vídeo y el bucket de Amazon S3 desde el que recuperarlo. `NotificationChannel` contiene el Nombre de recurso de Amazon (ARN) del tema de Amazon SNS al que Amazon Rekognition Video notifica cuando finaliza la solicitud de análisis de vídeo. El tema de Amazon SNS debe estar en la misma región de AWS que el punto de conexión de Amazon Rekognition Video al que está llamando. `NotificationChannel` también contiene el ARN de un rol que permite a Amazon Rekognition Video publicar en el tema de Amazon SNS. Puede conceder permisos de publicación a Amazon Rekognition a sus temas de creando un rol de servicio de IAM. Para obtener más información, consulte Configuración de Amazon Rekognition Video.

También puede especificar un parámetro de entrada opcional, `JobTag`, que le permite identificar el trabajo en el estado de realización que se ha publicado en el tema de Amazon SNS.

Para evitar la duplicación accidental de trabajos de análisis, tiene la opción de proporcionar un token idempotente, `ClientRequestToken`. Si proporciona un valor para `ClientRequestToken`, la operación `Start` devuelve el mismo `JobId` para varias llamadas idénticas a la operación de inicio, como por ejemplo `StartLabelDetection`. Un token `ClientRequestToken` tiene una vida útil de 7 días. Después de 7 días, puede volver a utilizarla. Si reutiliza el token durante el ciclo de vida del token, sucede lo siguiente:

- Si reutiliza el token con la misma operación `Start` y los mismos parámetros de entrada, se devuelve el mismo `JobId`. El trabajo no se vuelve a realizar de nuevo y Amazon Rekognition Video no envía un estado de realización al tema de Amazon SNS registrado.
- Si vuelve a utilizar el token con la misma operación `Start` y un cambio de parámetro de entrada menor, obtendrá una excepción `IdempotentParameterMismatchException` (código de estado HTTP: 400).
- No debe reutilizar un token con diferentes operaciones `Start`, ya que obtendrá resultados impredecibles en Amazon Rekognition.

La respuesta a la operación `StartLabelDetection` es un identificador de trabajo (`JobId`). Utilice `JobId` para realizar un seguimiento de las solicitudes y obtener los resultados de análisis después de que Amazon Rekognition Video haya publicado el estado de realización en el tema de Amazon SNS. Por ejemplo:

```
{"JobId":"270c1cc5e1d0ea2fbc59d97cb69a72a5495da75851976b14a1784ca90fc180e3"}
```

Si inicia demasiados trabajos al mismo tiempo, las llamadas a `StartLabelDetection` producen una excepción `LimitExceededException` (código de estado HTTP: 400) hasta que el número de trabajos ejecutados simultáneamente se encuentre por debajo del límite de servicio de Amazon Rekognition.

Si descubre que las excepciones `LimitExceededException` se producen con picos de actividad, considere la posibilidad de usar una cola de Amazon SQS para administrar las solicitudes entrantes. Póngase en contacto con el servicio de AWS asistencia si descubre que una cola de Amazon SQS no puede gestionar su número medio de solicitudes simultáneas y sigue recibiendo excepciones. `LimitExceededException`

Obtención del estado de realización de una solicitud de análisis de Amazon Rekognition Video

Amazon Rekognition Video envía una notificación a la realización de análisis al tema de Amazon SNS registrado. La notificación incluye el identificador de trabajo y el estado de realización de la operación en una cadena de JSON. Una solicitud de análisis de vídeo correcta tiene un estado `SUCCEEDED`. Por ejemplo, el siguiente resultado muestra el procesamiento correcto de un trabajo de detección de etiqueta.

```
{
    "JobId": "270c1cc5e1d0ea2fbc59d97cb69a72a5495da75851976b14a1nnnnnnnnnnn",
    "Status": "SUCCEEDED",
    "API": "StartLabelDetection",
    "JobTag": "DetectingLabels",
    "Timestamp": 1510865364756,
    "Video": {
        "S3ObjectName": "video.mp4",
        "S3Bucket": "amzn-s3-demo-bucket"
    }
}
```

Para obtener más información, consulte Referencia: notificación de resultados de análisis de vídeo.

Para obtener la información de estado que Amazon Rekognition Video ha publicado en el tema de Amazon SNS, utilice una de las siguientes opciones:

- AWS Lambda: puede suscribir una función de AWS Lambda que escriba en un tema de Amazon SNS. Se llama a la función cuando Amazon Rekognition notifica al tema de Amazon SNS que la solicitud se ha completado. Utilice una función de Lambda si desea que el código del servidor procese los resultados de una solicitud de análisis de vídeo. Por ejemplo, es posible que desee utilizar el código del servidor para anotar el vídeo o crear un informe sobre el contenido de vídeo antes de devolver la información a una aplicación cliente. También le recomendamos el procesamiento del lado del servidor de vídeos grandes, ya que la API de Amazon Rekognition podría devolver grandes volúmenes de datos.
- Amazon Simple Queue Service: puede suscribir una cola de Amazon SQS a un tema de Amazon SNS. A continuación, sondee la cola de Amazon SQS para recuperar el estado de realización que ha publicado Amazon Rekognition cuando se completa una solicitud de análisis de vídeo. Para obtener más información, consulte Análisis de un vídeo almacenado en un bucket de Amazon

S3 con Java o Python (SDK). Utilice una cola de Amazon SQS si desea llamar a operaciones de Amazon Rekognition Video solo desde una aplicación cliente.

> **Important**
>
> No le recomendamos obtener el estado de realización de solicitud llamando repetidamente a la operación `Get` de Amazon Rekognition Video. Esto se debe a que Amazon Rekognition Video limita la operación `Get` si se realizan demasiadas solicitudes. Si está procesando varios vídeos simultáneamente, es más sencillo y más eficaz monitorear una cola de SQS para la notificación de realización que sondear Amazon Rekognition Video para detectar el estado de cada vídeo individualmente.

Obtención de los resultados del análisis de Amazon Rekognition Video

Para obtener los resultados de una solicitud de análisis de vídeo, en primer lugar, asegúrese de que el estado de realización que se ha recuperado del tema de Amazon SNS es `SUCCEEDED`. A continuación, llame a `GetLabelDetection`, que transfiere el valor `JobId` que se devuelve desde `StartLabelDetection`. El JSON de la solicitud es similar al siguiente ejemplo:

```
{
    "JobId": "270c1cc5e1d0ea2fbc59d97cb69a72a5495da75851976b14a1784ca90fc180e3",
    "MaxResults": 10,
    "SortBy": "TIMESTAMP"
}
```

JobId es el identificador de la operación de análisis de vídeo. Dado que el análisis de vídeo puede generar grandes cantidades de datos, utilice `MaxResults` para especificar el número máximo de resultados que debe devolver en una sola operación GET. El valor predeterminado de `MaxResults` es 1000. Si especifica un valor superior a 1 000, se devolverá un máximo de 1 000 resultados. Si la operación no devuelve todo el conjunto de resultados, se devuelve un token de paginación para la página siguiente en la respuesta de operación. Si tiene un token de paginación de una solicitud GET anterior, utilícelo con `NextToken` para obtener la siguiente página de resultados.

> **Note**
>
> Amazon Rekognition retiene los resultados de una operación de análisis de vídeo durante siete días. No podrá recuperar los resultados del análisis transcurrido este plazo.

El JSON de respuesta de la operación `GetLabelDetection` es similar al siguiente:

```
{
    "Labels": [
        {
            "Timestamp": 0,
            "Label": {
                "Instances": [],
                "Confidence": 60.51791763305664,
                "Parents": [],
                "Name": "Electronics"
            }
        },
        {
            "Timestamp": 0,
            "Label": {
                "Instances": [],
                "Confidence": 99.53411102294922,
                "Parents": [],
                "Name": "Human"
            }
        },
        {
            "Timestamp": 0,
            "Label": {
                "Instances": [
                    {
                        "BoundingBox": {
                            "Width": 0.11109819263219833,
                            "Top": 0.08098889887332916,
                            "Left": 0.8881205320358276,
                            "Height": 0.9073750972747803
                        },
                        "Confidence": 99.5831298828125
                    },
                    {
                        "BoundingBox": {
```

```
                            "Width": 0.1268676072359085,
                            "Top": 0.14018426835536957,
                            "Left": 0.0003282368124928324,
                            "Height": 0.7993982434272766
                        },
                        "Confidence": 99.46029663085938
                    }
                ],
                "Confidence": 99.53411102294922,
                "Parents": [],
                "Name": "Person"
            }
        },
        .
        .
        .

        {
            "Timestamp": 166,
            "Label": {
                "Instances": [],
                "Confidence": 73.6471176147461,
                "Parents": [
                    {
                        "Name": "Clothing"
                    }
                ],
                "Name": "Sleeve"
            }
        }

    ],
    "LabelModelVersion": "2.0",
    "JobStatus": "SUCCEEDED",
    "VideoMetadata": {
        "Format": "QuickTime / MOV",
        "FrameRate": 23.976024627685547,
        "Codec": "h264",
        "DurationMillis": 5005,
        "FrameHeight": 674,
        "FrameWidth": 1280
    }
}
```

Las operaciones `GetLabelDetection` y `GetContentModeration` le permiten ordenar los resultados del análisis por marca de tiempo o por nombre de etiqueta. También puede agregar los resultados por segmento de vídeo o por marca de tiempo.

Puede ordenar los resultados por hora de detección (milisegundos desde el comienzo del vídeo) o alfabéticamente por la entidad detectada (objeto, rostro, famoso, etiqueta de moderación o persona). Para ordenar por tiempo, establezca el valor del parámetro de entrada `SortBy` en `TIMESTAMP`. Si no se especifica `SortBy`, el comportamiento predeterminado se ordena por tiempo. El ejemplo anterior está ordenado por tiempo. Para ordenar por entidad, utilice el parámetro de entrada `SortBy` con el valor que es adecuado para la operación que está realizando. Por ejemplo, para ordenar por etiqueta detectada en una llamada a `GetLabelDetection`, utilice el valor `NAME`.

Para agregar los resultados por marca de tiempo, defina el valor del parámetro `AggregateBy` en `TIMESTAMPS`. Para agregar por segmento de vídeo, defina el valor de `AggregateBy` en `SEGMENTS`. El modo de agregación de `SEGMENTS` agregará las etiquetas a lo largo del tiempo y `TIMESTAMPS` mostrará la marca temporal en la que se detectó una etiqueta, utilizando un muestreo de 2 FPS y una salida por fotograma (Nota: Esta frecuencia de muestreo actual está sujeta a cambios, por lo que no se deben hacer suposiciones sobre la frecuencia de muestreo actual). Si no se especifica ningún valor, el método de agregación predeterminado es `TIMESTAMPS`.

Configuración de Amazon Rekognition Video

Para utilizar la API de Amazon Rekognition Video con vídeos almacenados, debe configurar el usuario y un rol de servicio de IAM para que puedan tener acceso a los temas de Amazon SNS. También tiene que suscribir una cola de Amazon SQS a los temas de Amazon SNS.

> **Note**
>
> Si utiliza estas instrucciones para configurar el ejemplo de Análisis de un vídeo almacenado en un bucket de Amazon S3 con Java o Python (SDK), no es necesario que realice los pasos 3, 4, 5 y 6. El ejemplo contiene código que permite crear y configurar el tema de Amazon SNS y la cola de Amazon SQS.

Los ejemplos de esta sección crean un tema de Amazon SNS nuevo siguiendo las instrucciones que conceden a Amazon Rekognition Video acceso a varios temas. Si desea utilizar un tema de Amazon SNS existente, utilice Otorgar acceso a un tema de Amazon SNS existente en el paso 3.

Para configurar Amazon Rekognition Video

1. Configure una AWS cuenta para acceder a Amazon Rekognition Video. Para obtener más información, consulte Paso 1: Configurar una cuenta de AWS y crear un usuario.
2. Instale y configure el SDK necesario. AWS Para obtener más información, consulte Paso 2: Configure y AWS CLI AWS SDKs.
3. Para ejecutar los ejemplos de código de esta guía para desarrolladores, asegúrese de que el usuario elegido tenga acceso mediante programación. Para obtener más información, consulte Concesión de acceso programático.

   El usuario también necesita al menos los siguientes permisos:

   - Amazon SQSFull Access
   - AmazonRekognitionFullAccess
   - Amazon S3 FullAccess
   - Amazon SNSFull Access

   Si utiliza el Centro de identidades de IAM para autenticarse, añada los permisos al conjunto de permisos de su rol; de lo contrario, añada los permisos a su rol de IAM.
4. Cree un tema de Amazon SNS utilizando la consola de Amazon SNS. Añada al nombre del tema un prefijo. AmazonRekognition Anote el nombre de recurso de Amazon (ARN) del tema. Asegúrese de que el tema está situado en la misma región que el punto de conexión de AWS que está utilizando.
5. Cree una cola estándar de Amazon SQS mediante la consola de Amazon SQS. Anote el ARN de la cola.
6. Suscriba la cola al tema que creó en el paso 3.
7. Conceda permiso al tema de Amazon SNS y enviar mensajes a la cola de Amazon SQS.
8. Cree un rol de servicio de IAM para otorgar a Amazon Rekognition Video acceso a sus temas de Amazon SNS. Anote el nombre de recurso de Amazon (ARN) del rol de servicio. Para obtener más información, consulte Otorgar acceso a varios temas de Amazon SNS.
9. Para garantizar que su cuenta esté segura, querrá limitar el alcance del acceso de Rekognition solo a los recursos que esté utilizando. Para ello, puede adjuntar una política de confianza a su rol de servicio de IAM. Para obtener información sobre cómo hacerlo, consulte Prevención de la sustitución confusa entre servicios.
10. Añada la siguiente política insertada al usuario que ha creado en el paso 1:

Asigne a la política insertada un nombre de su elección.

11. Si utilizas una AWS Key Management Service clave gestionada por el cliente para cifrar los vídeos de tu bucket de Amazon S3, añade permisos a la clave para que el rol de servicio que creaste en el paso 7 descifre los vídeos. Como mínimo, el rol de servicio necesita permisos para las acciones `kms:GenerateDataKey` y `kms:Decrypt`. Por ejemplo:

```
{
    "Sid": "Decrypt only",
    "Effect": "Allow",
    "Principal": {
        "AWS": "arn:aws:iam::111122223333:user/user from step 1"
    },
    "Action": [
        "kms:Decrypt",
        "kms:GenerateDataKey"
    ],
    "Resource": "*"
}
```

Para obtener más información, consulte Mi bucket de Amazon S3 tiene un cifrado predeterminado mediante una clave de AWS KMS personalizada. ¿Cómo puedo permitir que los usuarios realicen operaciones de carga y descarga en el bucket? y Protección de los datos con el cifrado del lado del servidor con claves de KMS almacenadas en AWS Key Management Service (SSE-KMS).

12. Ahora puede ejecutar los ejemplos de Análisis de un vídeo almacenado en un bucket de Amazon S3 con Java o Python (SDK) y Analizar un vídeo con el AWS Command Line Interface.

Otorgar acceso a varios temas de Amazon SNS

Use un rol de servicio de IAM para otorgar a Amazon Rekognition Video acceso a los temas de Amazon SNS que cree. IAM proporciona el caso de uso Rekognition para la creación de un rol de servicio de Amazon Rekognition Video.

Puede conceder a Amazon Rekognition Video acceso a varios temas de Amazon SNS utilizando la política de permisos y anteponiendo los nombres de `AmazonRekognitionServiceRole` los temas con —por ejemplo,. AmazonRekognition`AmazonRekognitionMyTopicName`

Para dar acceso a Amazon Rekognition Video a varios temas de Amazon SNS

1. Cree un rol de servicio de IAM. Utilice la siguiente información para crear el rol de servicio de IAM:
    1. Elija Rekognition para el nombre del servicio.
    2. Elija Rekognition para el caso de uso del rol de servicio. Debería ver la política de permisos en la lista. AmazonRekognitionServiceRole AmazonRekognitionServiceRolepermite a Amazon Rekognition Video acceder a los temas de Amazon SNS con el prefijo. AmazonRekognition
    3. Asigne al rol de servicio un nombre de su elección.
2. Anote el ARN del rol de servicio. Lo necesita para comenzar las operaciones de análisis de vídeo.

Otorgar acceso a un tema de Amazon SNS existente

Puede crear una política de permisos que permita a Amazon Rekognition Video acceder a un tema de Amazon SNS.

Para dar acceso a Amazon Rekognition Video a un tema de Amazon SNS existente

1. Cree una nueva política de permisos con el editor de políticas de JSON de IAM y utilice la política siguiente. Reemplace `topicarn` por el nombre de recurso de Amazon (ARN) del tema de Amazon SNS deseado.
2. Cree un rol de servicio de IAM o actualice un rol de servicio de IAM existente. Utilice la siguiente información para crear el rol de servicio de IAM:
    1. Elija Rekognition para el nombre del servicio.
    2. Elija Rekognition para el caso de uso del rol de servicio.
    3. Adjunte la política de permisos que ha creado en el paso 1.
3. Anote el ARN del rol de servicio. Lo necesita para comenzar las operaciones de análisis de vídeo.

Análisis de un vídeo almacenado en un bucket de Amazon S3 con Java o Python (SDK)

Este procedimiento muestra cómo detectar etiquetas en un vídeo mediante la utilización de las operaciones de detección de etiquetas de Amazon Rekognition Video, un vídeo almacenado en un bucket de Amazon S3 y un tema de Amazon SNS. El procedimiento también muestra cómo utilizar una cola de Amazon SQS para obtener el estado de realización del tema de Amazon SNS. Para obtener más información, consulte Cómo llamar a las operaciones de Amazon Rekognition Video. No está limitado a la utilización de una cola de Amazon SQS: Por ejemplo, puede usar una AWS Lambda función para obtener el estado de finalización. Para obtener más información, consulte Invocación de funciones Lambda mediante notificaciones de Amazon SNS.

En el código de ejemplo de este procedimiento, se explica cómo hacer lo siguiente:

1. Cree el tema de Amazon SNS.
2. Cree la cola de Amazon SQS.
3. Dar a Amazon Rekognition Video permiso para publicar el estado de realización de una operación de análisis de vídeo en el tema de Amazon SNS.
4. Suscriba la cola de Amazon SQS al tema de Amazon SNS.
5. Inicie la solicitud de análisis de vídeo llamando a StartLabelDetection.
6. Obtenga el estado de realización a partir de la cola de Amazon SQS. En el ejemplo se hace un seguimiento del identificador de trabajo (`JobId`) que devuelve `StartLabelDetection` y solo obtiene los resultados de identificadores de trabajo coincidentes que se leen desde el estado de realización. Se trata de un aspecto importante si otras aplicaciones están utilizando la misma cola y tema. Para simplificar, en el ejemplo se eliminan los trabajos que no coinciden. Plantéese añadirlos a una cola de mensajes fallidos de Amazon SQS para examinarlos posteriormente.
7. Obtenga y muestre los resultados del análisis de vídeo llamando a GetLabelDetection.

Requisitos previos

El código de ejemplo de este procedimiento se proporciona en Java y Python. Debe tener instalado el AWS SDK correspondiente. Para obtener más información, consulte Introducción a Amazon Rekognition. La cuenta de AWS que utilice debe tener permisos de acceso a la API de Amazon Rekognition. Para obtener más información, consulte Acciones definidas por Amazon Rekognition.

Para detectar etiquetas en un vídeo

1. Configure el acceso de los usuarios a Amazon Rekognition Video y configure el acceso de Amazon Rekognition Video a Amazon SNS. Para obtener más información, consulte Configuración de Amazon Rekognition Video. No es necesario realizar los pasos 3, 4, 5 y 6, ya que el código de ejemplo crea y configura el tema de Amazon SNS y la cola de Amazon SQS.
2. Cargue un archivo de vídeo con formato MOV o MPEG-4 en el bucket de Amazon S3. Para realizar pruebas, cargue un vídeo con una duración inferior a 30 segundos.

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

3. Utilice los siguientes ejemplos de código para detectar etiquetas en un vídeo.

   Java

   En la función `main`:

   - reemplace `roleArn` por el ARN del rol de servicio de IAM que creó en el paso 7 de Para configurar Amazon Rekognition Video.
   - Reemplace los valores de `amzn-s3-demo-bucket` y `video` por el bucket y el nombre del archivo de vídeo que especificó en el paso 2.

```
//Copyright 2018 Amazon.com, Inc. or its affiliates. All Rights Reserved.
//PDX-License-Identifier: MIT-0 (For details, see https://github.com/awsdocs/
amazon-rekognition-developer-guide/blob/master/LICENSE-SAMPLECODE.)

package com.amazonaws.samples;
import com.amazonaws.auth.policy.Policy;
import com.amazonaws.auth.policy.Condition;
import com.amazonaws.auth.policy.Principal;
import com.amazonaws.auth.policy.Resource;
import com.amazonaws.auth.policy.Statement;
import com.amazonaws.auth.policy.Statement.Effect;
import com.amazonaws.auth.policy.actions.SQSActions;
import com.amazonaws.services.rekognition.AmazonRekognition;
import com.amazonaws.services.rekognition.AmazonRekognitionClientBuilder;
import com.amazonaws.services.rekognition.model.CelebrityDetail;
import com.amazonaws.services.rekognition.model.CelebrityRecognition;
import com.amazonaws.services.rekognition.model.CelebrityRecognitionSortBy;
```

```
import com.amazonaws.services.rekognition.model.ContentModerationDetection;
import com.amazonaws.services.rekognition.model.ContentModerationSortBy;
import com.amazonaws.services.rekognition.model.Face;
import com.amazonaws.services.rekognition.model.FaceDetection;
import com.amazonaws.services.rekognition.model.FaceMatch;
import com.amazonaws.services.rekognition.model.FaceSearchSortBy;
import com.amazonaws.services.rekognition.model.GetCelebrityRecognitionRequest;
import com.amazonaws.services.rekognition.model.GetCelebrityRecognitionResult;
import com.amazonaws.services.rekognition.model.GetContentModerationRequest;
import com.amazonaws.services.rekognition.model.GetContentModerationResult;
import com.amazonaws.services.rekognition.model.GetFaceDetectionRequest;
import com.amazonaws.services.rekognition.model.GetFaceDetectionResult;
import com.amazonaws.services.rekognition.model.GetFaceSearchRequest;
import com.amazonaws.services.rekognition.model.GetFaceSearchResult;
import com.amazonaws.services.rekognition.model.GetLabelDetectionRequest;
import com.amazonaws.services.rekognition.model.GetLabelDetectionResult;
import com.amazonaws.services.rekognition.model.GetPersonTrackingRequest;
import com.amazonaws.services.rekognition.model.GetPersonTrackingResult;
import com.amazonaws.services.rekognition.model.Instance;
import com.amazonaws.services.rekognition.model.Label;
import com.amazonaws.services.rekognition.model.LabelDetection;
import com.amazonaws.services.rekognition.model.LabelDetectionSortBy;
import com.amazonaws.services.rekognition.model.NotificationChannel;
import com.amazonaws.services.rekognition.model.Parent;
import com.amazonaws.services.rekognition.model.PersonDetection;
import com.amazonaws.services.rekognition.model.PersonMatch;
import com.amazonaws.services.rekognition.model.PersonTrackingSortBy;
import com.amazonaws.services.rekognition.model.S3Object;
import
 com.amazonaws.services.rekognition.model.StartCelebrityRecognitionRequest;
import com.amazonaws.services.rekognition.model.StartCelebrityRecognitionResult;
import com.amazonaws.services.rekognition.model.StartContentModerationRequest;
import com.amazonaws.services.rekognition.model.StartContentModerationResult;
import com.amazonaws.services.rekognition.model.StartFaceDetectionRequest;
import com.amazonaws.services.rekognition.model.StartFaceDetectionResult;
import com.amazonaws.services.rekognition.model.StartFaceSearchRequest;
import com.amazonaws.services.rekognition.model.StartFaceSearchResult;
import com.amazonaws.services.rekognition.model.StartLabelDetectionRequest;
import com.amazonaws.services.rekognition.model.StartLabelDetectionResult;
import com.amazonaws.services.rekognition.model.StartPersonTrackingRequest;
import com.amazonaws.services.rekognition.model.StartPersonTrackingResult;
import com.amazonaws.services.rekognition.model.Video;
import com.amazonaws.services.rekognition.model.VideoMetadata;
import com.amazonaws.services.sns.AmazonSNS;
```

```
import com.amazonaws.services.sns.AmazonSNSClientBuilder;
import com.amazonaws.services.sns.model.CreateTopicRequest;
import com.amazonaws.services.sns.model.CreateTopicResult;
import com.amazonaws.services.sqs.AmazonSQS;
import com.amazonaws.services.sqs.AmazonSQSClientBuilder;
import com.amazonaws.services.sqs.model.CreateQueueRequest;
import com.amazonaws.services.sqs.model.Message;
import com.amazonaws.services.sqs.model.QueueAttributeName;
import com.amazonaws.services.sqs.model.SetQueueAttributesRequest;
import com.fasterxml.jackson.databind.JsonNode;
import com.fasterxml.jackson.databind.ObjectMapper;
import java.util.*;

public class VideoDetect {


    private static String sqsQueueName=null;
    private static String snsTopicName=null;
    private static String snsTopicArn = null;
    private static String roleArn= null;
    private static String sqsQueueUrl = null;
    private static String sqsQueueArn = null;
    private static String startJobId = null;
    private static String bucket = null;
    private static String video = null;
    private static AmazonSQS sqs=null;
    private static AmazonSNS sns=null;
    private static AmazonRekognition rek = null;

    private static NotificationChannel channel= new NotificationChannel()
            .withSNSTopicArn(snsTopicArn)
            .withRoleArn(roleArn);


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

        video = "";
        bucket = "";
        roleArn= "";

        sns = AmazonSNSClientBuilder.defaultClient();
        sqs= AmazonSQSClientBuilder.defaultClient();
        rek = AmazonRekognitionClientBuilder.defaultClient();
```

```
        CreateTopicandQueue();

        //=================================================

        StartLabelDetection(bucket, video);

        if (GetSQSMessageSuccess()==true)
         GetLabelDetectionResults();

       //=================================================


        DeleteTopicandQueue();
        System.out.println("Done!");

    }


    static boolean GetSQSMessageSuccess() throws Exception
    {
     boolean success=false;


        System.out.println("Waiting for job: " + startJobId);
        //Poll queue for messages
        List<Message> messages=null;
        int dotLine=0;
        boolean jobFound=false;

        //loop until the job status is published. Ignore other messages in
queue.
        do{
            messages = sqs.receiveMessage(sqsQueueUrl).getMessages();
            if (dotLine++<40){
                System.out.print(".");
            }else{
                System.out.println();
                dotLine=0;
            }

            if (!messages.isEmpty()) {
                //Loop through messages received.
                for (Message message: messages) {
                    String notification = message.getBody();
```

```
                    // Get status and job id from notification.
                    ObjectMapper mapper = new ObjectMapper();
                    JsonNode jsonMessageTree = mapper.readTree(notification);
                    JsonNode messageBodyText = jsonMessageTree.get("Message");
                    ObjectMapper operationResultMapper = new ObjectMapper();
                    JsonNode jsonResultTree =
operationResultMapper.readTree(messageBodyText.textValue());
                    JsonNode operationJobId = jsonResultTree.get("JobId");
                    JsonNode operationStatus = jsonResultTree.get("Status");
                    System.out.println("Job found was " + operationJobId);
                    // Found job. Get the results and display.
                    if(operationJobId.asText().equals(startJobId)){
                        jobFound=true;
                        System.out.println("Job id: " + operationJobId );
                        System.out.println("Status : " +
operationStatus.toString());
                        if (operationStatus.asText().equals("SUCCEEDED")){
                         success=true;
                        }
                        else{
                            System.out.println("Video analysis failed");
                        }


sqs.deleteMessage(sqsQueueUrl,message.getReceiptHandle());
                    }

                    else{
                        System.out.println("Job received was not job " +
startJobId);
                        //Delete unknown message. Consider moving message to
dead letter queue

sqs.deleteMessage(sqsQueueUrl,message.getReceiptHandle());
                    }
                }
            }
            else {
                Thread.sleep(5000);
            }
        } while (!jobFound);

        System.out.println("Finished processing video");
```

```
        return success;
    }


    private static void StartLabelDetection(String bucket, String video) throws
Exception{

        NotificationChannel channel= new NotificationChannel()
                .withSNSTopicArn(snsTopicArn)
                .withRoleArn(roleArn);


        StartLabelDetectionRequest req = new StartLabelDetectionRequest()
                .withVideo(new Video()
                        .withS3Object(new S3Object()
                                .withBucket(bucket)
                                .withName(video)))
                .withMinConfidence(50F)
                .withJobTag("DetectingLabels")
                .withNotificationChannel(channel);

        StartLabelDetectionResult startLabelDetectionResult =
rek.startLabelDetection(req);
        startJobId=startLabelDetectionResult.getJobId();

    }

    private static void GetLabelDetectionResults() throws Exception{

        int maxResults=10;
        String paginationToken=null;
        GetLabelDetectionResult labelDetectionResult=null;

        do {
            if (labelDetectionResult !=null){
                paginationToken = labelDetectionResult.getNextToken();
            }

            GetLabelDetectionRequest labelDetectionRequest= new
GetLabelDetectionRequest()
                    .withJobId(startJobId)
                    .withSortBy(LabelDetectionSortBy.TIMESTAMP)
                    .withMaxResults(maxResults)
                    .withNextToken(paginationToken);
```

```
            labelDetectionResult = rek.getLabelDetection(labelDetectionRequest);

            VideoMetadata videoMetaData=labelDetectionResult.getVideoMetadata();

            System.out.println("Format: " + videoMetaData.getFormat());
            System.out.println("Codec: " + videoMetaData.getCodec());
            System.out.println("Duration: " +
videoMetaData.getDurationMillis());
            System.out.println("FrameRate: " + videoMetaData.getFrameRate());


            //Show labels, confidence and detection times
            List<LabelDetection> detectedLabels=
labelDetectionResult.getLabels();

            for (LabelDetection detectedLabel: detectedLabels) {
                long seconds=detectedLabel.getTimestamp();
                Label label=detectedLabel.getLabel();
                System.out.println("Millisecond: " + Long.toString(seconds) + "
");

                System.out.println("   Label:" + label.getName());
                System.out.println("   Confidence:" +
detectedLabel.getLabel().getConfidence().toString());

                List<Instance> instances = label.getInstances();
                System.out.println("   Instances of " + label.getName());
                if (instances.isEmpty()) {
                    System.out.println("        " + "None");
                } else {
                    for (Instance instance : instances) {
                        System.out.println("        Confidence: " +
instance.getConfidence().toString());
                        System.out.println("        Bounding box: " +
instance.getBoundingBox().toString());
                    }
                }
                System.out.println("   Parent labels for " + label.getName() +
":");
                List<Parent> parents = label.getParents();
                if (parents.isEmpty()) {
                    System.out.println("        None");
```

```
                } else {
                    for (Parent parent : parents) {
                        System.out.println("        " + parent.getName());
                    }
                }
                System.out.println();
            }
        } while (labelDetectionResult !=null &&
labelDetectionResult.getNextToken() != null);

    }

    // Creates an SNS topic and SQS queue. The queue is subscribed to the
topic.
    static void CreateTopicandQueue()
    {
        //create a new SNS topic
        snsTopicName="AmazonRekognitionTopic" +
Long.toString(System.currentTimeMillis());
        CreateTopicRequest createTopicRequest = new
CreateTopicRequest(snsTopicName);
        CreateTopicResult createTopicResult =
sns.createTopic(createTopicRequest);
        snsTopicArn=createTopicResult.getTopicArn();

        //Create a new SQS Queue
        sqsQueueName="AmazonRekognitionQueue" +
Long.toString(System.currentTimeMillis());
        final CreateQueueRequest createQueueRequest = new
CreateQueueRequest(sqsQueueName);
        sqsQueueUrl = sqs.createQueue(createQueueRequest).getQueueUrl();
        sqsQueueArn = sqs.getQueueAttributes(sqsQueueUrl,
Arrays.asList("QueueArn")).getAttributes().get("QueueArn");

        //Subscribe SQS queue to SNS topic
        String sqsSubscriptionArn = sns.subscribe(snsTopicArn, "sqs",
sqsQueueArn).getSubscriptionArn();

        // Authorize queue
          Policy policy = new Policy().withStatements(
                  new Statement(Effect.Allow)
                  .withPrincipals(Principal.AllUsers)
                  .withActions(SQSActions.SendMessage)
                  .withResources(new Resource(sqsQueueArn))
```

```
                    .withConditions(new
Condition().withType("ArnEquals").withConditionKey("aws:SourceArn").withValues(snsTopic
                    );


            Map queueAttributes = new HashMap();
            queueAttributes.put(QueueAttributeName.Policy.toString(),
policy.toJson());
            sqs.setQueueAttributes(new SetQueueAttributesRequest(sqsQueueUrl,
queueAttributes));


           System.out.println("Topic arn: " + snsTopicArn);
           System.out.println("Queue arn: " + sqsQueueArn);
           System.out.println("Queue url: " + sqsQueueUrl);
           System.out.println("Queue sub arn: " + sqsSubscriptionArn );
        }
       static void DeleteTopicandQueue()
       {
           if (sqs !=null) {
               sqs.deleteQueue(sqsQueueUrl);
               System.out.println("SQS queue deleted");
           }

           if (sns!=null) {
               sns.deleteTopic(snsTopicArn);
               System.out.println("SNS topic deleted");
           }
       }
}
```

Python

En la función `main`:

- reemplace `roleArn` por el ARN del rol de servicio de IAM que creó en el paso 7 de Para configurar Amazon Rekognition Video.
- Reemplace los valores de `amzn-s3-demo-bucket` y `video` por el bucket y el nombre del archivo de vídeo que especificó en el paso 2.
- Sustituya el valor de `profile_name` en la línea que crea la sesión de Rekognition por el nombre de su perfil de desarrollador.

- También puede incluir criterios de filtrado en el parámetro de configuración. Por ejemplo, puede utilizar un `LabelsInclusionFilter` o un `LabelsExclusionFilter` junto a una lista de los valores deseados. En el código que aparece a continuación, puede eliminar los comentarios de las secciones `Features` y `Settings` y proporcionar sus propios valores para limitar los resultados devueltos solo a las etiquetas que le interesen.
- En la llamada a `GetLabelDetection`, puede proporcionar valores para los argumentos `SortBy` y `AggregateBy`. Para ordenar por tiempo, establezca el valor del parámetro de entrada `SortBy` en `TIMESTAMP`. Para ordenar por entidad, utilice el parámetro de entrada `SortBy` con el valor que es adecuado para la operación que está realizando. Para agregar los resultados por marca de tiempo, defina el valor del parámetro `AggregateBy` en `TIMESTAMPS`. Para agregar por segmento de vídeo, utilice `SEGMENTS`.

En el ejemplo de código, coloque la siguiente política de IAM entre las comillas de la línea `policy = """""".format(sqsQueueArn, self.snsTopicArn)`.

```
## Copyright 2018 Amazon.com, Inc. or its affiliates. All Rights Reserved.
# PDX-License-Identifier: MIT-0 (For details, see https://github.com/awsdocs/
amazon-rekognition-developer-guide/blob/master/LICENSE-SAMPLECODE.)

import boto3
import json
import sys
import time

class VideoDetect:

    jobId = ''

    roleArn = ''
    bucket = ''
    video = ''
    startJobId = ''

    sqsQueueUrl = ''
    snsTopicArn = ''
    processType = ''

    def __init__(self, role, bucket, video, client, rek, sqs, sns):
        self.roleArn = role
```

```
        self.bucket = bucket
        self.video = video
        self.client = client
        self.rek = rek
        self.sqs = sqs
        self.sns = sns

    def GetSQSMessageSuccess(self):

        jobFound = False
        succeeded = False

        dotLine = 0
        while jobFound == False:
            sqsResponse = self.sqs.receive_message(QueueUrl=self.sqsQueueUrl,
MessageAttributeNames=['ALL'],
                                                   MaxNumberOfMessages=10)

            if sqsResponse:

                if 'Messages' not in sqsResponse:
                    if dotLine < 40:
                        print('.', end='')
                        dotLine = dotLine + 1
                    else:
                        print()
                        dotLine = 0
                    sys.stdout.flush()
                    time.sleep(5)
                    continue

                for message in sqsResponse['Messages']:
                    notification = json.loads(message['Body'])
                    rekMessage = json.loads(notification['Message'])
                    print(rekMessage['JobId'])
                    print(rekMessage['Status'])
                    if rekMessage['JobId'] == self.startJobId:
                        print('Matching Job Found:' + rekMessage['JobId'])
                        jobFound = True
                        if (rekMessage['Status'] == 'SUCCEEDED'):
                            succeeded = True

                        self.sqs.delete_message(QueueUrl=self.sqsQueueUrl,
```

```
ReceiptHandle=message['ReceiptHandle'])
                    else:
                        print("Job didn't match:" +
                              str(rekMessage['JobId']) + ' : ' +
self.startJobId)
                    # Delete the unknown message. Consider sending to dead
letter queue
                    self.sqs.delete_message(QueueUrl=self.sqsQueueUrl,

ReceiptHandle=message['ReceiptHandle'])

        return succeeded

    def StartLabelDetection(self):
        response = self.rek.start_label_detection(Video={'S3Object': {'Bucket':
self.bucket, 'Name': self.video}},

NotificationChannel={'RoleArn': self.roleArn,

'SNSTopicArn': self.snsTopicArn},
                                                  MinConfidence=90,
                                                  # Filtration options,
uncomment and add desired labels to filter returned labels
                                                  # Features=['GENERAL_LABELS'],
                                                  # Settings={
                                                  # 'GeneralLabels': {
                                                  # 'LabelInclusionFilters':
['Clothing']
                                                  # }}
                                                   )

        self.startJobId = response['JobId']
        print('Start Job Id: ' + self.startJobId)

    def GetLabelDetectionResults(self):
        maxResults = 10
        paginationToken = ''
        finished = False

        while finished == False:
            response = self.rek.get_label_detection(JobId=self.startJobId,
                                                    MaxResults=maxResults,
                                                    NextToken=paginationToken,
```

```
                                                    SortBy='TIMESTAMP',
                                                    AggregateBy="TIMESTAMPS")

            print('Codec: ' + response['VideoMetadata']['Codec'])
            print('Duration: ' + str(response['VideoMetadata']
['DurationMillis']))
            print('Format: ' + response['VideoMetadata']['Format'])
            print('Frame rate: ' + str(response['VideoMetadata']['FrameRate']))
            print()

            for labelDetection in response['Labels']:
                label = labelDetection['Label']

                print("Timestamp: " + str(labelDetection['Timestamp']))
                print("   Label: " + label['Name'])
                print("   Confidence: " + str(label['Confidence']))
                print("   Instances:")
                for instance in label['Instances']:
                    print("      Confidence: " + str(instance['Confidence']))
                    print("      Bounding box")
                    print("        Top: " + str(instance['BoundingBox']['Top']))
                    print("        Left: " + str(instance['BoundingBox']
['Left']))
                    print("        Width: " + str(instance['BoundingBox']
['Width']))
                    print("        Height: " + str(instance['BoundingBox']
['Height']))
                    print()
                print()

                print("Parents:")
                for parent in label['Parents']:
                    print("   " + parent['Name'])

                print("Aliases:")
                for alias in label['Aliases']:
                    print("   " + alias['Name'])

                print("Categories:")
                for category in label['Categories']:
                    print("   " + category['Name'])
                print("----------")
                print()
```

```
                if 'NextToken' in response:
                    paginationToken = response['NextToken']
                else:
                    finished = True

    def CreateTopicandQueue(self):

        millis = str(int(round(time.time() * 1000)))

        # Create SNS topic

        snsTopicName = "AmazonRekognitionExample" + millis

        topicResponse = self.sns.create_topic(Name=snsTopicName)
        self.snsTopicArn = topicResponse['TopicArn']

        # create SQS queue
        sqsQueueName = "AmazonRekognitionQueue" + millis
        self.sqs.create_queue(QueueName=sqsQueueName)
        self.sqsQueueUrl = self.sqs.get_queue_url(QueueName=sqsQueueName)
['QueueUrl']

        attribs = self.sqs.get_queue_attributes(QueueUrl=self.sqsQueueUrl,
                                                AttributeNames=['QueueArn'])
['Attributes']

        sqsQueueArn = attribs['QueueArn']

        # Subscribe SQS queue to SNS topic
        self.sns.subscribe(
            TopicArn=self.snsTopicArn,
            Protocol='sqs',
            Endpoint=sqsQueueArn)

        # Authorize SNS to write SQS queue
        # Insert the policy from the documentation from
        policy = """
        {{
           "Version":"2012-10-17",
           "Statement":[
            {{
                "Sid":"MyPolicy",
                "Effect":"Allow",
                "Principal" : {{"AWS" : "*"}},
```

```
                "Action":"SQS:SendMessage",
                "Resource": "{}",
                "Condition":{{
                    "ArnEquals":{{
                        "aws:SourceArn": "{}"
                     }}
                }}
            }}
            ]
        }}""".format(sqsQueueArn, self.snsTopicArn)

        response = self.sqs.set_queue_attributes(
            QueueUrl=self.sqsQueueUrl,
            Attributes={
                'Policy': policy
            })

    def DeleteTopicandQueue(self):
        self.sqs.delete_queue(QueueUrl=self.sqsQueueUrl)
        self.sns.delete_topic(TopicArn=self.snsTopicArn)

def main():

    roleArn = 'role-arn'
    bucket = 'bucket-name'
    video = 'video-name'

    session = boto3.Session(profile_name='profile-name')
    client = session.client('rekognition')
    rek = boto3.client('rekognition')
    sqs = boto3.client('sqs')
    sns = boto3.client('sns')

    analyzer = VideoDetect(roleArn, bucket, video, client, rek, sqs, sns)
    analyzer.CreateTopicandQueue()

    analyzer.StartLabelDetection()
    if analyzer.GetSQSMessageSuccess() == True:
        analyzer.GetLabelDetectionResults()

    analyzer.DeleteTopicandQueue()

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

Node.Js

En el siguiente código de muestra:

- Sustituya el valor de `REGION` por el nombre de la región operativa de su cuenta.
- Sustituya el valor de `amzn-s3-demo-bucket` por el nombre del bucket de Amazon S3 que contiene el archivo de vídeo.
- Sustituya el valor de `videoName` por el nombre del archivo de vídeo en su bucket de Amazon S3.
- Sustituya el valor de `profile_name` en la línea que crea la sesión de Rekognition por el nombre de su perfil de desarrollador.
- reemplace `roleArn` por el ARN del rol de servicio de IAM que creó en el paso 7 de Para configurar Amazon Rekognition Video.

```
import { CreateQueueCommand, GetQueueAttributesCommand, GetQueueUrlCommand,
  SetQueueAttributesCommand, DeleteQueueCommand, ReceiveMessageCommand,
 DeleteMessageCommand } from  "@aws-sdk/client-sqs";
import {CreateTopicCommand, SubscribeCommand, DeleteTopicCommand } from "@aws-
sdk/client-sns";
import  { SQSClient } from "@aws-sdk/client-sqs";
import  { SNSClient } from "@aws-sdk/client-sns";
import  { RekognitionClient, StartLabelDetectionCommand,
 GetLabelDetectionCommand } from "@aws-sdk/client-rekognition";
import { stdout } from "process";
import {fromIni} from '@aws-sdk/credential-providers';

// Set the AWS Region.
const REGION = "region-name"; //e.g. "us-east-1"
const profileName = "profile-name"
// Create SNS service object.
const sqsClient = new SQSClient({ region: REGION,
  credentials: fromIni({profile: profileName,}), });
const snsClient = new SNSClient({ region: REGION,
  credentials: fromIni({profile: profileName,}), });
const rekClient = new RekognitionClient({region: REGION,
  credentials: fromIni({profile: profileName,}),
});
```

```
// Set bucket and video variables
const bucket = "bucket-name";
const videoName = "video-name";
const roleArn = "role-arn"
var startJobId = ""

var ts = Date.now();
const snsTopicName = "AmazonRekognitionExample" + ts;
const snsTopicParams = {Name: snsTopicName}
const sqsQueueName = "AmazonRekognitionQueue-" + ts;

// Set the parameters
const sqsParams = {
  QueueName: sqsQueueName, //SQS_QUEUE_URL
  Attributes: {
    DelaySeconds: "60", // Number of seconds delay.
    MessageRetentionPeriod: "86400", // Number of seconds delay.
  },
};

const createTopicandQueue = async () => {
  try {
    // Create SNS topic
    const topicResponse = await snsClient.send(new
 CreateTopicCommand(snsTopicParams));
    const topicArn = topicResponse.TopicArn
    console.log("Success", topicResponse);
    // Create SQS Queue
    const sqsResponse = await sqsClient.send(new
 CreateQueueCommand(sqsParams));
    console.log("Success", sqsResponse);
    const sqsQueueCommand = await sqsClient.send(new
 GetQueueUrlCommand({QueueName: sqsQueueName}))
    const sqsQueueUrl = sqsQueueCommand.QueueUrl
    const attribsResponse = await sqsClient.send(new
 GetQueueAttributesCommand({QueueUrl: sqsQueueUrl, AttributeNames:
 ['QueueArn']}))
    const attribs = attribsResponse.Attributes
    console.log(attribs)
    const queueArn = attribs.QueueArn
    // subscribe SQS queue to SNS topic
    const subscribed = await snsClient.send(new SubscribeCommand({TopicArn:
 topicArn, Protocol:'sqs', Endpoint: queueArn}))
    const policy = {
```

```
      Version: "2012-10-17",&TCX5-2025-waiver;
      Statement: [
        {
          Sid: "MyPolicy",
          Effect: "Allow",
          Principal: {AWS: "*"},
          Action: "SQS:SendMessage",
          Resource: queueArn,
          Condition: {
            ArnEquals: {
              'aws:SourceArn': topicArn
            }
          }
        }
      ]
    };

    const response = sqsClient.send(new SetQueueAttributesCommand({QueueUrl:
 sqsQueueUrl, Attributes: {Policy: JSON.stringify(policy)}}))
    console.log(response)
    console.log(sqsQueueUrl, topicArn)
    return [sqsQueueUrl, topicArn]

  } catch (err) {
    console.log("Error", err);
  }
};

const startLabelDetection = async (roleArn, snsTopicArn) => {
  try {
    //Initiate label detection and update value of startJobId with returned Job
 ID
   const labelDetectionResponse = await rekClient.send(new
 StartLabelDetectionCommand({Video:{S3Object:{Bucket:bucket, Name:videoName}},
      NotificationChannel:{RoleArn: roleArn, SNSTopicArn: snsTopicArn}}));
      startJobId = labelDetectionResponse.JobId
      console.log(`JobID: ${startJobId}`)
      return startJobId
  } catch (err) {
    console.log("Error", err);
  }
};

const getLabelDetectionResults = async(startJobId) => {
```

```
 console.log("Retrieving Label Detection results")
 // Set max results, paginationToken and finished will be updated depending on
response values
 var maxResults = 10
 var paginationToken = ''
 var finished = false

 // Begin retrieving label detection results
 while (finished == false){
   var response = await rekClient.send(new GetLabelDetectionCommand({JobId:
startJobId, MaxResults: maxResults,
     NextToken: paginationToken, SortBy:'TIMESTAMP'}))
     // Log metadata
     console.log(`Codec: ${response.VideoMetadata.Codec}`)
     console.log(`Duration: ${response.VideoMetadata.DurationMillis}`)
     console.log(`Format: ${response.VideoMetadata.Format}`)
     console.log(`Frame Rate: ${response.VideoMetadata.FrameRate}`)
     console.log()
     // For every detected label, log label, confidence, bounding box, and
timestamp
     response.Labels.forEach(labelDetection => {
       var label = labelDetection.Label
       console.log(`Timestamp: ${labelDetection.Timestamp}`)
       console.log(`Label: ${label.Name}`)
       console.log(`Confidence: ${label.Confidence}`)
       console.log("Instances:")
       label.Instances.forEach(instance =>{
         console.log(`Confidence: ${instance.Confidence}`)
         console.log("Bounding Box:")
         console.log(`Top: ${instance.Confidence}`)
         console.log(`Left: ${instance.Confidence}`)
         console.log(`Width: ${instance.Confidence}`)
         console.log(`Height: ${instance.Confidence}`)
         console.log()
       })
     console.log()
     // Log parent if found
     console.log("   Parents:")
     label.Parents.forEach(parent =>{
       console.log(`    ${parent.Name}`)
     })
     console.log()
     // Searh for pagination token, if found, set variable to next token
     if (String(response).includes("NextToken")){
```

```
        paginationToken = response.NextToken

      }else{
        finished = true
      }

      })
  }
}

// Checks for status of job completion
const getSQSMessageSuccess = async(sqsQueueUrl, startJobId) => {
  try {
    // Set job found and success status to false initially
    var jobFound = false
    var succeeded = false
    var dotLine = 0
    // while not found, continue to poll for response
    while (jobFound == false){
      var sqsReceivedResponse = await sqsClient.send(new
 ReceiveMessageCommand({QueueUrl:sqsQueueUrl,
        MaxNumberOfMessages:'ALL', MaxNumberOfMessages:10}));
      if (sqsReceivedResponse){
        var responseString = JSON.stringify(sqsReceivedResponse)
        if (!responseString.includes('Body')){
          if (dotLine < 40) {
            console.log('.')
            dotLine = dotLine + 1
          }else {
            console.log('')
            dotLine = 0
          };
          stdout.write('', () => {
            console.log('');
          });
          await new Promise(resolve => setTimeout(resolve, 5000));
          continue
        }
      }

      // Once job found, log Job ID and return true if status is succeeded
      for (var message of sqsReceivedResponse.Messages){
        console.log("Retrieved messages:")
        var notification = JSON.parse(message.Body)
```

```
        var rekMessage = JSON.parse(notification.Message)
        var messageJobId = rekMessage.JobId
        if (String(rekMessage.JobId).includes(String(startJobId))){
          console.log('Matching job found:')
          console.log(rekMessage.JobId)
          jobFound = true
          console.log(rekMessage.Status)
          if (String(rekMessage.Status).includes(String("SUCCEEDED"))){
            succeeded = true
            console.log("Job processing succeeded.")
            var sqsDeleteMessage = await sqsClient.send(new
 DeleteMessageCommand({QueueUrl:sqsQueueUrl,
 ReceiptHandle:message.ReceiptHandle}));
          }
        }else{
          console.log("Provided Job ID did not match returned ID.")
          var sqsDeleteMessage = await sqsClient.send(new
 DeleteMessageCommand({QueueUrl:sqsQueueUrl,
 ReceiptHandle:message.ReceiptHandle}));
        }
      }
    }
  return succeeded
  } catch(err) {
    console.log("Error", err);
  }
};

// Start label detection job, sent status notification, check for success
 status
// Retrieve results if status is "SUCEEDED", delete notification queue and
 topic
const runLabelDetectionAndGetResults = async () => {
  try {
    const sqsAndTopic = await createTopicandQueue();
    const startLabelDetectionRes = await startLabelDetection(roleArn,
 sqsAndTopic[1]);
    const getSQSMessageStatus = await getSQSMessageSuccess(sqsAndTopic[0],
 startLabelDetectionRes)
    console.log(getSQSMessageSuccess)
    if (getSQSMessageSuccess){
      console.log("Retrieving results:")
      const results = await getLabelDetectionResults(startLabelDetectionRes)
    }
```

```
    const deleteQueue = await sqsClient.send(new DeleteQueueCommand({QueueUrl:
 sqsAndTopic[0]}));
    const deleteTopic = await snsClient.send(new DeleteTopicCommand({TopicArn:
 sqsAndTopic[1]}));
    console.log("Successfully deleted.")
  } catch (err) {
    console.log("Error", err);
  }
};

runLabelDetectionAndGetResults()
```

Java V2

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

```
import com.fasterxml.jackson.core.JsonProcessingException;
import com.fasterxml.jackson.databind.JsonMappingException;
import com.fasterxml.jackson.databind.JsonNode;
import com.fasterxml.jackson.databind.ObjectMapper;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import
 software.amazon.awssdk.services.rekognition.model.StartLabelDetectionResponse;
import software.amazon.awssdk.services.rekognition.model.NotificationChannel;
import software.amazon.awssdk.services.rekognition.model.S3Object;
import software.amazon.awssdk.services.rekognition.model.Video;
import
 software.amazon.awssdk.services.rekognition.model.StartLabelDetectionRequest;
import
 software.amazon.awssdk.services.rekognition.model.GetLabelDetectionRequest;
import
 software.amazon.awssdk.services.rekognition.model.GetLabelDetectionResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import software.amazon.awssdk.services.rekognition.model.LabelDetectionSortBy;
import software.amazon.awssdk.services.rekognition.model.VideoMetadata;
import software.amazon.awssdk.services.rekognition.model.LabelDetection;
import software.amazon.awssdk.services.rekognition.model.Label;
import software.amazon.awssdk.services.rekognition.model.Instance;
import software.amazon.awssdk.services.rekognition.model.Parent;
import software.amazon.awssdk.services.sqs.SqsClient;
```

```
import software.amazon.awssdk.services.sqs.model.Message;
import software.amazon.awssdk.services.sqs.model.ReceiveMessageRequest;
import software.amazon.awssdk.services.sqs.model.DeleteMessageRequest;
import java.util.List;
//snippet-end:[rekognition.java2.recognize_video_detect.import]

/**
* Before running this Java V2 code example, set up your development environment,
 including your credentials.
*
* For more information, see the following documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
*/
public class VideoDetect {

 private static String startJobId ="";
 public static void main(String[] args) {

     final String usage = "\n" +
         "Usage: " +
         "   <bucket> <video> <queueUrl> <topicArn> <roleArn>\n\n" +
         "Where:\n" +
         "   bucket - The name of the bucket in which the video is located (for
 example, (for example, amzn-s3-demo-bucket). \n\n"+
         "   video - The name of the video (for example, people.mp4). \n\n" +
         "   queueUrl- The URL of a SQS queue. \n\n" +
         "   topicArn - The ARN of the Amazon Simple Notification Service
 (Amazon SNS) topic. \n\n" +
         "   roleArn - The ARN of the AWS Identity and Access Management (IAM)
 role to use. \n\n" ;

     if (args.length != 5) {
         System.out.println(usage);
         System.exit(1);
     }

     String bucket = args[0];
     String video = args[1];
     String queueUrl = args[2];
     String topicArn = args[3];
     String roleArn = args[4];
     Region region = Region.US_WEST_2;
```

```
    RekognitionClient rekClient = RekognitionClient.builder()
        .region(region)
        .credentialsProvider(ProfileCredentialsProvider.create("profile-name"))
        .build();

    SqsClient sqs = SqsClient.builder()
        .region(Region.US_WEST_2)
        .credentialsProvider(ProfileCredentialsProvider.create("profile-name"))
        .build();

    NotificationChannel channel = NotificationChannel.builder()
        .snsTopicArn(topicArn)
        .roleArn(roleArn)
        .build();

    startLabels(rekClient, channel, bucket, video);
    getLabelJob(rekClient, sqs,  queueUrl);
    System.out.println("This example is done!");
    sqs.close();
    rekClient.close();
}

// snippet-start:[rekognition.java2.recognize_video_detect.main]
public static void startLabels(RekognitionClient rekClient,
                               NotificationChannel channel,
                               String bucket,
                               String video) {
    try {
        S3Object s3Obj = S3Object.builder()
            .bucket(bucket)
            .name(video)
            .build();

        Video vidOb = Video.builder()
            .s3Object(s3Obj)
            .build();

        StartLabelDetectionRequest labelDetectionRequest =
StartLabelDetectionRequest.builder()
            .jobTag("DetectingLabels")
            .notificationChannel(channel)
            .video(vidOb)
            .minConfidence(50F)
            .build();
```

```
        StartLabelDetectionResponse labelDetectionResponse =
rekClient.startLabelDetection(labelDetectionRequest);
        startJobId = labelDetectionResponse.jobId();

        boolean ans = true;
        String status = "";
        int yy = 0;
        while (ans) {

            GetLabelDetectionRequest detectionRequest =
GetLabelDetectionRequest.builder()
                .jobId(startJobId)
                .maxResults(10)
                .build();

            GetLabelDetectionResponse result =
rekClient.getLabelDetection(detectionRequest);
            status = result.jobStatusAsString();

            if (status.compareTo("SUCCEEDED") == 0)
                ans = false;
            else
                System.out.println(yy +" status is: "+status);

            Thread.sleep(1000);
            yy++;
        }

        System.out.println(startJobId +" status is: "+status);

    } catch(RekognitionException | InterruptedException e) {
        e.getMessage();
        System.exit(1);
    }
}

public static void getLabelJob(RekognitionClient rekClient, SqsClient sqs,
String queueUrl) {

    List<Message> messages;
    ReceiveMessageRequest messageRequest = ReceiveMessageRequest.builder()
        .queueUrl(queueUrl)
        .build();
```

```
    try {
        messages = sqs.receiveMessage(messageRequest).messages();

        if (!messages.isEmpty()) {
            for (Message message: messages) {
                String notification = message.body();

                // Get the status and job id from the notification
                ObjectMapper mapper = new ObjectMapper();
                JsonNode jsonMessageTree = mapper.readTree(notification);
                JsonNode messageBodyText = jsonMessageTree.get("Message");
                ObjectMapper operationResultMapper = new ObjectMapper();
                JsonNode jsonResultTree =
operationResultMapper.readTree(messageBodyText.textValue());
                JsonNode operationJobId = jsonResultTree.get("JobId");
                JsonNode operationStatus = jsonResultTree.get("Status");
                System.out.println("Job found in JSON is " + operationJobId);

                DeleteMessageRequest deleteMessageRequest =
DeleteMessageRequest.builder()
                    .queueUrl(queueUrl)
                    .build();

                String jobId = operationJobId.textValue();
                if (startJobId.compareTo(jobId)==0) {
                    System.out.println("Job id: " + operationJobId );
                    System.out.println("Status : " +
operationStatus.toString());

                    if (operationStatus.asText().equals("SUCCEEDED"))
                        GetResultsLabels(rekClient);
                    else
                        System.out.println("Video analysis failed");

                    sqs.deleteMessage(deleteMessageRequest);
                }

                else{
                    System.out.println("Job received was not job " +
startJobId);
                    sqs.deleteMessage(deleteMessageRequest);
                }
            }
```

```
        }

    } catch(RekognitionException e) {
        e.getMessage();
        System.exit(1);
    } catch (JsonMappingException e) {
        e.printStackTrace();
    } catch (JsonProcessingException e) {
        e.printStackTrace();
    }
}

// Gets the job results by calling GetLabelDetection
private static void GetResultsLabels(RekognitionClient rekClient) {

    int maxResults=10;
    String paginationToken=null;
    GetLabelDetectionResponse labelDetectionResult=null;

    try {
        do {
            if (labelDetectionResult !=null)
                paginationToken = labelDetectionResult.nextToken();


            GetLabelDetectionRequest labelDetectionRequest=
GetLabelDetectionRequest.builder()
                .jobId(startJobId)
                .sortBy(LabelDetectionSortBy.TIMESTAMP)
                .maxResults(maxResults)
                .nextToken(paginationToken)
                .build();

            labelDetectionResult =
rekClient.getLabelDetection(labelDetectionRequest);
            VideoMetadata videoMetaData=labelDetectionResult.videoMetadata();
            System.out.println("Format: " + videoMetaData.format());
            System.out.println("Codec: " + videoMetaData.codec());
            System.out.println("Duration: " + videoMetaData.durationMillis());
            System.out.println("FrameRate: " + videoMetaData.frameRate());

            List<LabelDetection> detectedLabels= labelDetectionResult.labels();
            for (LabelDetection detectedLabel: detectedLabels) {
                long seconds=detectedLabel.timestamp();
```

```
                Label label=detectedLabel.label();
                System.out.println("Millisecond: " + seconds + " ");

                System.out.println("   Label:" + label.name());
                System.out.println("   Confidence:" +
 detectedLabel.label().confidence().toString());

                List<Instance> instances = label.instances();
                System.out.println("   Instances of " + label.name());

                if (instances.isEmpty()) {
                    System.out.println("        " + "None");
                } else {
                    for (Instance instance : instances) {
                        System.out.println("        Confidence: " +
 instance.confidence().toString());
                        System.out.println("        Bounding box: " +
 instance.boundingBox().toString());
                    }
                }
                System.out.println("   Parent labels for " + label.name() +
 ":");
                List<Parent> parents = label.parents();

                if (parents.isEmpty()) {
                    System.out.println("        None");
                } else {
                    for (Parent parent : parents) {
                        System.out.println("   " + parent.name());
                    }
                }
                System.out.println();
            }
        } while (labelDetectionResult !=null &&
 labelDetectionResult.nextToken() != null);

    } catch(RekognitionException e) {
        e.getMessage();
        System.exit(1);
    }
 }
 // snippet-end:[rekognition.java2.recognize_video_detect.main]
}
```

4. Cree y ejecute el código. La operación podría llevar algún tiempo. Una vez terminada, se muestra una lista de las etiquetas detectadas en el vídeo. Para obtener más información, consulte Detección de etiquetas en un vídeo.

Analizar un vídeo con el AWS Command Line Interface

Puede usar AWS Command Line Interface (AWS CLI) para llamar a las operaciones de Amazon Rekognition Video. El patrón de diseño es el mismo que cuando se utiliza la API Amazon Rekognition Video con AWS u otro dispositivo. AWS SDK para Java SDKs Para obtener más información, consulte Descripción general de la API de Amazon Rekognition Video. Los siguientes procedimientos muestran cómo utilizarlos AWS CLI para detectar etiquetas en un vídeo.

Comienza la detección de etiquetas en un vídeo llamando a `start-label-detection`. Cuando Amazon Rekognition termina de analizar el vídeo, el estado de realización se envía al tema de Amazon SNS que se ha especificado en el parámetro `--notification-channel` de `start-label-detection`. Puede obtener el estado de realización suscribiendo una cola de Amazon Simple Queue Service (Amazon SQS) al tema de Amazon SNS. A continuación, sondee receive-message para obtener el estado de realización de la cola de Amazon SQS.

Al llamar a `StartLabelDetection`, puede filtrar los resultados proporcionando argumentos de filtrado a los argumentos `LabelsInclusionFilter` y/o `LabelsExclusionFilter`. Para obtener más información, consulte Detección de etiquetas en un vídeo .

La notificación del estado de realización es una estructura de JSON dentro de la respuesta `receive-message`. Tiene que extraer el JSON de la respuesta. Para obtener más información acerca del JSON del estado de realización, consulte Referencia: notificación de resultados de análisis de vídeo. Si el valor del campo `Status` del JSON de estado completado es `SUCCEEDED`, puede obtener los resultados de la solicitud de análisis llamando a `get-label-detection`. Al llamar a `GetLabelDetection`, puede ordenar y agregar los resultados devueltos utilizando los argumentos `SortBy` y `AggregateBy`.

Los siguientes procedimientos no incluyen código para sondear la cola de Amazon SQS. Además, no incluyen código para analizar el JSON que se devuelve desde la cola de Amazon SQS. Para ver un ejemplo en Java, consulte Análisis de un vídeo almacenado en un bucket de Amazon S3 con Java o Python (SDK).

Requisitos previos

Para ejecutar este procedimiento, debe tener el AWS CLI instalado. Para obtener más información, consulte Introducción a Amazon Rekognition. La cuenta de AWS que utilice debe tener permisos de acceso a la API de Amazon Rekognition. Para obtener más información, Acciones definidas por Amazon Rekognition.

Para configurar Amazon Rekognition Video y subir un vídeo

1. Configure el acceso de los usuarios a Amazon Rekognition Video y configure el acceso de Amazon Rekognition Video a Amazon SNS. Para obtener más información, consulte Configuración de Amazon Rekognition Video.
2. Cargue un archivo de vídeo con formato MOV o MPEG-4 en el bucket de S3. Para desarrollo y pruebas, le aconsejamos que utilice vídeos cortos con una duración inferior a 30 segundos.

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

Para detectar etiquetas en un vídeo

1. Ejecute el siguiente AWS CLI comando para empezar a detectar etiquetas en un vídeo.

```
aws rekognition start-label-detection --video '{"S3Object":{"Bucket":"amzn-s3-demo-
bucket","Name":"video-name"}}' \
 --notification-channel '{"SNSTopicArn":"TopicARN","RoleArn":"RoleARN"}' \
--region region-name  \
--features GENERAL_LABELS \
--profile profile-name \
--settings "{"GeneralLabels":{"LabelInclusionFilters":["Car"]}}
```

   Actualice los siguientes valores:

   - Cambie `amzn-s3-demo-bucket` y `videofile` por el nombre del bucket de Amazon S3 y el nombre de archivo que especificó en el paso 2.
   - Cambie `us-east-1` por la región de AWS que está utilizando.
   - Sustituya el valor de `profile_name` en la línea que crea la sesión de Rekognition por el nombre de su perfil de desarrollador.

- Reemplace `TopicARN` por el ARN del tema de Amazon SNS que creó en el paso 3 de Configuración de Amazon Rekognition Video.
- Cambie `RoleARN` por el ARN del rol de servicio de IAM que creó en el paso 7 de Configuración de Amazon Rekognition Video.
- Si es necesario, puede especificar la `endpoint-url`. La CLI de AWS debe determinar automáticamente la URL del punto de conexión adecuada en función de la región proporcionada. Sin embargo, si utiliza un punto de conexión de su VPC privada, es posible que deba especificar la `endpoint-url`. El recurso AWS Service Endpoints incluye la sintaxis para especificar las direcciones URL de los puntos de conexión y los nombres y códigos de cada región.
- También puede incluir criterios de filtrado en el parámetro de configuración. Por ejemplo, puede utilizar un `LabelsInclusionFilter` o un `LabelsExclusionFilter` junto a una lista de los valores deseados.

Si accede a la CLI en un dispositivo Windows, utilice comillas dobles en lugar de comillas simples y evite las comillas dobles internas con una barra invertida (es decir, \) para corregir cualquier error del analizador que pueda encontrar. Consulte a continuación un ejemplo:

```
aws rekognition start-label-detection --video "{\"S3Object\":{\"Bucket\":\"amzn-s3-
demo-bucket",\"Name\":\"video-name\"}}" --notification-channel "{\"SNSTopicArn\":
\"TopicARN\",\"RoleArn\":\"RoleARN\"}" \
--region us-east-1 --features GENERAL_LABELS --settings "{\"GeneralLabels\":
{\"LabelInclusionFilters\":[\"Car\"]}}" --profile profile-name
```

2. Anote el valor de `JobId` en la respuesta. La respuesta tiene un aspecto similar a la del siguiente ejemplo JSON.

```
{
    "JobId": "547089ce5b9a8a0e7831afa655f42e5d7b5c838553f1a584bf350ennnnnnnnnn"
}
```

3. Escriba código para sondear la cola de Amazon SQS para ver el JSON del estado de realización (mediante la utilización de receive-message).
4. Escriba código para extraer el campo `Status` del JSON de estado de realización.

5. Si el valor `Status` es`SUCCEEDED`, ejecute el siguiente AWS CLI comando para mostrar los resultados de la detección de etiquetas.

```
aws rekognition get-label-detection  --job-id JobId \
--region us-east-1 --sort-by TIMESTAMP aggregate-by TIMESTAMPS
```

Actualice los siguientes valores:

- Cambie `JobId` para que coincida con el identificador de trabajo que ha anotado en el paso 2.
- Cambie `Endpoint` y `us-east-1` al punto de conexión y la región de AWS que está utilizando.

Los resultados tienen un aspecto similar al JSON del siguiente ejemplo:

```
{
    "Labels": [
        {
            "Timestamp": 0,
            "Label": {
                "Confidence": 99.03720092773438,
                "Name": "Speech"
            }
        },
        {
            "Timestamp": 0,
            "Label": {
                "Confidence": 71.6698989868164,
                "Name": "Pumpkin"
            }
        },
        {
            "Timestamp": 0,
            "Label": {
                "Confidence": 71.6698989868164,
                "Name": "Squash"
            }
        },
        {
            "Timestamp": 0,
            "Label": {
                "Confidence": 71.6698989868164,
```

```
                "Name": "Vegetable"
            }
        }, .......
```

Referencia: notificación de resultados de análisis de vídeo

Amazon Rekognition publica los resultados de una solicitud de análisis de Amazon Rekognition Video, incluido el estado de finalización, en un tema de Amazon Simple Notification Service (Amazon SNS). Para obtener la notificación de un tema de Amazon SNS, utilice una cola o una función de Amazon Simple Queue Service. AWS Lambda Para obtener más información, consulte the section called “Cómo llamar a las operaciones de Amazon Rekognition Video”. Para ver un ejemplo, consulta Análisis de un vídeo almacenado en un bucket de Amazon S3 con Java o Python (SDK).

La carga está en el siguiente formato JSON:

```
{
  "JobId": "String",
  "Status": "String",
  "API": "String",
  "JobTag": "String",
  "Timestamp": Number,
  "Video": {
    "S3ObjectName": "String",
    "S3Bucket": "String"
  }
}
```

Name	Description (Descripción)
JobId	El identificador del trabajo. Coincide con un identificador de trabajo que se devuelve desde una operación `Start`, como por ejemplo StartPersonTracking.
Status	El estado del trabajo. Los valores válidos son SUCCEEDED, FAILED o ERROR.
API	La operación de Amazon Rekognition Video utilizada para analizar el vídeo de entrada.

Name	Description (Descripción)
JobTag	Identificador del trabajo. Especifique `JobTag` en una llamada a una operación Start, como por ejemplo StartLabelDetection.
Timestamp	La marca de hora Unix cuando el trabajo terminó.
Video	Detalles sobre el vídeo que se procesó. Incluye el nombre de archivo y el bucket de Amazon S3 en el que está almacenado el archivo.

A continuación se muestra un ejemplo de una notificación de éxito que se envió a un tema de Amazon SNS.

```
{
  "JobId": "6de014b0-2121-4bf0-9e31-856a18719e22",
  "Status": "SUCCEEDED",
  "API": "LABEL_DETECTION",
  "Message": "",
  "Timestamp": 1502230160926,
  "Video": {
    "S3ObjectName": "video.mpg",
    "S3Bucket": "amzn-s3-demo-bucket"
  }
}
```

Solución de problemas de Amazon Rekognition Video

En este tema se incluye información de solución de problemas relacionados con el uso de Amazon Rekognition Video y de vídeos almacenados.

No recibo nunca el estado de realización que se envía al tema de Amazon SNS

Amazon Rekognition Video publica información de estado en un tema de Amazon SNS cuando finaliza el análisis de vídeo. Por lo general, el mensaje de estado de realización se obtiene al suscribirse al tema mediante una cola de Amazon SQS o una función de Lambda. Para ayudarle en su investigación, suscríbase al tema de Amazon SNS por correo electrónico para recibir los

mensajes que se envían al tema de Amazon SNS en su bandeja de entrada. Para obtener más información, consulte Suscripción a un tema de Amazon SNS.

Si no recibe el mensaje en la aplicación, haga lo siguiente:

- Compruebe que el análisis ha finalizado. Compruebe el valor de `JobStatus` en la respuesta de la operación GET (por ejemplo, `GetLabelDetection`). Si el valor es `IN_PROGRESS`, el análisis no ha finalizado, y el estado de realización todavía no se ha publicado en el tema de Amazon SNS.
- Compruebe que tiene un rol de servicio de IAM que concede a Amazon Rekognition Video permisos para publicar en los temas de Amazon SNS. Para obtener más información, consulte Configuración de Amazon Rekognition Video.
- Confirme que el rol de servicio de IAM que está utilizando puede publicarse en el tema de Amazon SNS mediante credenciales de rol y que los permisos de su rol de servicio están sujetos de forma segura a los recursos que está utilizando. Realice los pasos siguientes:
  - Obtenga el nombre de recurso de Amazon (ARN) del usuario:

    ```
    aws sts get-caller-identity --profile RekognitionUser
    ```

  - Añada el ARN del usuario a la relación de confianza del rol. Para obtener más información, consulte Modificación de un rol. El siguiente ejemplo de política de confianza especifica las credenciales del rol del usuario y restringe los permisos del rol de servicio solo a los recursos que está utilizando (para obtener más información sobre cómo limitar de forma segura el alcance de los permisos de un rol de servicio, consulte Prevención de la sustitución confusa entre servicios):
  - Asuma el rol: `aws sts assume-role --role-arn arn:Role ARN --role-session-name SessionName --profile RekognitionUser`
  - Publique en un tema de Amazon SNS: `aws sns publish --topic-arn arn:Topic ARN --message "Hello World!" --region us-east-1 --profile RekognitionUser`

  Si el comando AWS CLI funciona, recibirá el mensaje (en la bandeja de entrada del correo electrónico, si se ha suscrito al tema por correo electrónico). Si no recibe el mensaje:

  - Asegúrese de haber configurado Amazon Rekognition Video. Para obtener más información, consulte Configuración de Amazon Rekognition Video.
  - Asegúrese de que ha seguido los demás consejos que se ofrecen para esta pregunta.
- Compruebe que está utilizando el tema de Amazon SNS correcto:

- Si utiliza un rol de servicio de IAM para conceder a Amazon Rekognition Video acceso a un único tema de Amazon SNS, compruebe que ha dado permisos para el tema de Amazon SNS correcto. Para obtener más información, consulte Otorgar acceso a un tema de Amazon SNS existente.
- Si utiliza un rol de servicio de IAM para dar acceso a Amazon Rekognition Video a varios temas de SNS, compruebe que está utilizando el tema correcto y que el nombre del tema va precedido de él. AmazonRekognition Para obtener más información, consulte Otorgar acceso a varios temas de Amazon SNS.
- Si utiliza una AWS Lambda función, confirme que su función Lambda esté suscrita al tema de Amazon SNS correcto. Para obtener más información, consulte Distribución ramificada a las funciones de Lambda.

- Si suscribe una cola de Amazon SQS al tema de Amazon SNS, compruebe que el tema de Amazon SNS tiene permisos para enviar mensajes a la cola de Amazon SQS. Para obtener más información, consulte Dar permiso al tema de Amazon SNS para enviar mensajes a la cola de Amazon SQS.

Necesito ayuda adicional para solucionar el tema de Amazon SNS

Puede usarlo AWS X-Ray con Amazon SNS para rastrear y analizar los mensajes que viajan a través de su aplicación. Para obtener más información, consulte Amazon SNS y. AWS X-Ray

Para obtener ayuda adicional, puede publicar su pregunta en el foro de Amazon Rekognition o considerar la posibilidad de registrarse para recibir asistencia técnica de AWS.

Trabajar con eventos de vídeo en streaming

Puede usar Amazon Rekognition Video para detectar y reconocer rostros o detectar objetos en vídeo en streaming. Amazon Rekognition Video utiliza Amazon Kinesis Video Streams para recibir y procesar una transmisión de vídeo. Usted crea un procesador de transmisión con parámetros que muestran lo que desea que el procesador de transmisión detecte en la transmisión de vídeo. Rekognition envía los resultados de detección de etiquetas de eventos de vídeo en streaming como notificaciones de Amazon SNS y Amazon S3. Rekognition envía los resultados de la búsqueda de rostros a un flujo de datos de Kinesis.

Los procesadores de transmisión de búsqueda de rostros utilizan `FaceSearchSettings` para buscar rostros de una colección. Para obtener más información sobre cómo implementar los

procesadores de secuencias de búsqueda de rostros para analizar los rostros en la transmisión de vídeo, consulte the section called “Búsqueda de rostros en una colección en streaming de vídeo”.

Los procesadores de flujo de detección de etiquetas utilizan `ConnectedHomeSettings` para buscar personas, paquetes y mascotas en eventos de transmisión de vídeo. Para obtener más información acerca de cómo implementar los procesadores de flujo de detección de etiquetas, consulte the section called “Detección de etiquetas en eventos de vídeo en streaming”.

Descripción general de las operaciones del procesador de transmisión de Amazon Rekognition Video

El análisis de un vídeo en streaming comienza iniciando un procesador de streaming de Amazon Rekognition Video y transmitiendo vídeo a Amazon Rekognition Video. Un procesador de streaming de Amazon Rekognition Video le permite iniciar, detener y administrar procesadores de streaming. Crea un procesador de streaming llamando a CreateStreamProcessor. Los parámetros de solicitud para crear un procesador de secuencias de búsqueda de rostros incluyen los nombres de recursos de Amazon (ARNs) para la transmisión de vídeo de Kinesis, la transmisión de datos de Kinesis y el identificador de la colección que se utiliza para reconocer los rostros en la transmisión de vídeo. Los parámetros de solicitud para crear un procesador de transmisiones de monitoreo de seguridad incluyen los nombres de los recursos de Amazon (ARNs) para la transmisión de vídeo de Kinesis y el tema Amazon SNS, los tipos de objetos que desea detectar en la transmisión de vídeo y la información de un bucket de Amazon S3 para los resultados de salida. También incluye el nombre que especifica para el procesador de streaming.

Comienza a procesar un vídeo llamando a la operación StartStreamProcessor. Para obtener la información de estado de un procesador de streaming, llame a DescribeStreamProcessor. Otras operaciones a las que puede recurrir son TagResourceetiquetar un procesador de transmisión y DeleteStreamProcessoreliminar un procesador de transmisión. Si está utilizando un procesador de flujo con búsqueda facial, también puede usarlo StopStreamProcessorpara detener un procesador de flujo. Para obtener una lista de los procesadores de streaming en su cuenta, llame a ListStreamProcessors.

Después de que el procesador de streaming comienza a ejecutarse, transmita el vídeo en Amazon Rekognition Video a través de la transmisión de vídeo de Kinesis que especificó en `CreateStreamProcessor`. Puede utilizar la operación del PutMediaSDK de Kinesis Video Streams para entregar vídeo a la transmisión de vídeo de Kinesis. Para ver un ejemplo, consulte Ejemplo de PutMediaAPI.

Para obtener información sobre cómo su aplicación puede consumir los resultados del análisis de Amazon Rekognition Video de un procesador de secuencias de búsqueda facial, consulte Lectura de los resultados del análisis de vídeo en streaming.

Agregar etiquetas a un procesador de transmisión nuevo

Puede usar etiquetas para identificar, organizar, buscar y filtrar sus procesadores de transmisión de Amazon Rekognition mediante etiquetas. Cada etiqueta es una marca que consta de una clave y un valor definidos por el usuario.

Puede agregar etiquetas a un procesador de transmisión a medida que lo crea mediante la operación `CreateStreamProcessor`. Indique una o varias etiquetas en el parámetro de entrada de la matriz `Tags`. A continuación, se muestra un ejemplo de JSON para la solicitud `CreateStreamProcessor` con etiquetas.

```
{
       "Name": "streamProcessorForCam",
       "Input": {
              "KinesisVideoStream": {
                     "Arn": "arn:aws:kinesisvideo:us-east-1:nnnnnnnnnnnn:stream/
inputVideo"
              }
       },
       "Output": {
              "KinesisDataStream": {
                     "Arn": "arn:aws:kinesis:us-east-1:nnnnnnnnnnnn:stream/outputData"
              }
       },
       "RoleArn": "arn:aws:iam::nnnnnnnnnnn:role/roleWithKinesisPermission",
       "Settings": {
              "FaceSearch": {
                     "CollectionId": "collection-with-100-faces",
                     "FaceMatchThreshold": 85.5
              },
              "Tags": {
      "Dept": "Engineering",
        "Name": "Ana Silva Carolina",
        "Role": "Developer"
       }
}
```

Agregar etiquetas a un procesador de transmisión existente

Puede usar etiquetas para identificar, organizar, buscar y filtrar sus procesadores de transmisión de Amazon Rekognition mediante etiquetas. Cada etiqueta es una marca que consta de una clave y un valor definidos por el usuario.

Para agregar una o varias etiquetas a un procesador de transmisión, utilice la operación `TagResource`. Indique el nombre de recurso de Amazon (ARN) del procesador de transmisión (`ResourceArn`) y las etiquetas (`Tags`) que desea añadir. En el siguiente ejemplo se ve cómo añadir dos etiquetas.

```
aws rekognition tag-resource --resource-arn resource-arn \
                --tags '{"key1":"value1","key2":"value2"}'
```

> **Note**
>
> Si no conoce el nombre del recurso de Amazon del procesador de transmisión, puede usar la operación `DescribeStreamProcessor`.

Listar etiquetas en un procesador de flujo

Puede usar etiquetas para identificar, organizar, buscar y filtrar sus procesadores de transmisión de Amazon Rekognition mediante etiquetas. Cada etiqueta es una marca que consta de una clave y un valor definidos por el usuario.

Para enumerar las etiquetas adjuntas a un procesador de transmisión, utilice la operación `ListTagsForResource` y especifique el ARN del procesador de transmisión (`ResourceArn`). El resultado será la asignación de las claves y los valores de las etiquetas que se asocian al procesador de transmisión concreto.

```
aws rekognition list-tags-for-resource --resource-arn resource-arn
```

El resultado muestra una lista de etiquetas adjuntas al procesador de transmisión:

```
{
    "Tags": {
        "Dept": "Engineering",
        "Name": "Ana Silva Carolina",
        "Role": "Developer"
    }
}
```

Eliminar etiquetas de un procesador de transmisión

Puede usar etiquetas para identificar, organizar, buscar y filtrar sus procesadores de transmisión de Amazon Rekognition mediante etiquetas. Cada etiqueta es una marca que consta de una clave y un valor definidos por el usuario.

Para eliminar una o más etiquetas de un procesador de transmisión, utilice la operación `UntagResource`. Indique el ARN del modelo (`ResourceArn`) y las claves de etiqueta (`Tag-Keys`) que desee eliminar.

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

Si lo prefiere, también puede indicar claves de etiqueta en este formato:

```
--tag-keys key1,key2
```

Gestión de errores

En esta sección se describen los errores de tiempo de ejecución y se explica cómo controlarlos. También se describen los mensajes y códigos de error específicos de Amazon Rekognition.

Temas

- Componentes de un error
- Mensajes y códigos de error
- Control de errores en la aplicación

Componentes de un error

Cuando el programa envía una solicitud, Amazon Rekognition intenta procesarla. Si la solicitud se lleva a cabo correctamente, Amazon Rekognition devuelve un código de estado HTTP de operación correcta (`200 OK`), así como el resultado de la operación solicitada.

Si la solicitud no se realiza correctamente, Amazon Rekognition devuelve un error. Cada error tiene tres componentes:

- Un código de estado HTTP (por ejemplo, `400`).
- Un nombre de excepción (por ejemplo, `InvalidS3ObjectException`).
- Un mensaje de error (por ejemplo, `Unable to get object metadata from S3. Check object key, region and/or access permissions.`).

AWS se SDKs encarga de propagar los errores a su aplicación para que pueda tomar las medidas adecuadas. Por ejemplo, en un programa en Java, puede escribir una lógica `try-catch` para controlar una excepción `ResourceNotFoundException`.

Si no utiliza un SDK de AWS, tiene que analizar el contenido de la respuesta de bajo nivel de Amazon Rekognition. A continuación se muestra un ejemplo de este tipo de respuesta:

```
HTTP/1.1 400 Bad Request
Content-Type: application/x-amz-json-1.1
Date: Sat, 25 May 2019 00:28:25 GMT
x-amzn-RequestId: 03507c9b-7e84-11e9-9ad1-854a4567eb71
Content-Length: 222
Connection: keep-alive

{"__type":"InvalidS3ObjectException","Code":"InvalidS3ObjectException","Logref":"5022229e-7e48-
 to get object metadata from S3. Check object key, region and/or access permissions."}
```

Mensajes y códigos de error

A continuación se muestra una lista de excepciones que devuelve Amazon Rekognition, agrupados por su código de estado HTTP. Si ¿Reintentar? es Sí, puede volver a enviar la misma solicitud. Si ¿Reintentar? es No, debe corregir el problema en el lado del cliente antes de volver a enviar la solicitud.

Código de estado HTTP 400

Un código de estado HTTP `400` indica un problema con su solicitud. Algunos ejemplos de problemas son errores de autenticación, parámetros requeridos que faltan o que superan el rendimiento aprovisionado de una operación. Debe corregir el problema en la aplicación antes de volver a enviar la solicitud.

AccessDeniedException

Mensaje: Se produjo un error (AccessDeniedException) al llamar a la operación: El usuario: no está autorizado a realizar: en el recurso:. <Operation><User ARN><Operation><Resource ARN>

No tiene autorización para realizar la acción. Utilice el nombre de recurso de Amazon (ARN) de un usuario autorizado o un rol de IAM para realizar la operación.

¿Reintentar? No

GroupFacesInProgressException

Mensaje: No se pudo programar el GroupFaces trabajo. Existe un trabajo de agrupación de rostros para esta colección.

Vuelva a intentar la operación después de que finalice el trabajo existente.

¿Reintentar? No

IdempotentParameterMismatchException

Mensaje: El ClientRequestToken: que ha suministrado ya está en uso. <Token>

Se reutilizó un parámetro de ClientRequestToken entrada con una operación, pero al menos uno de los demás parámetros de entrada es diferente al de la llamada anterior a la operación.

¿Reintentar? No

ImageTooLargeException

Mensaje: El tamaño de la imagen es demasiado grande.

La imagen de entrada tamaño supera el límite permitido. Si llama DetectProtectiveEquipment, el tamaño o la resolución de la imagen superan el límite permitido. Para obtener más información, consulte Directrices y cuotas en Amazon Rekognition.

¿Reintentar? No

InvalidImageFormatException

Mensaje: La solicitud tiene un formato de imagen no válido.

No se admite el formato de imagen proporcionado. Utilice un formato de imagen admitido (.JPEG y.PNG). Para obtener más información, consulte Directrices y cuotas en Amazon Rekognition.

¿Reintentar? No

InvalidPaginationTokenException

Mensajes

- Token no válido
- Token de paginación no válido

El token de paginación de la solicitud no es válido. El token podría estar caducado.

¿Reintentar? No

InvalidParameterException

Mensaje: La solicitud tiene parámetros no válidos.

Un parámetro de entrada infringió una restricción. Valide los parámetros antes de llamar a la operación de la API de nuevo.

¿Reintentar? No

S3 no válido ObjectException

Mensajes:

- La solicitud tiene un objeto de S3 no válido.

- No se pueden obtener los metadatos de objeto desde S3. Compruebe la clave del objeto y los permisos de and/or acceso a la región.

Amazon Rekognition no puede obtener acceso al objeto de S3 que se especificó en la solicitud. Para obtener más información, consulte Configuración del acceso a S3: administración del acceso de AWS S3. Para obtener información sobre la solución de problemas, consulte Solución de problemas de Amazon S3.

¿Reintentar? No

LimitExceededException

Mensajes:

- Se ha superado el límite de procesador de flujo para la cuenta, límite: <límite actual>
- <Número de trabajos abiertos> trabajos abiertos para el usuario <ARN de usuario>, límite máximo: <límite máximo>

Se ha superado el límite de servicio de Amazon Rekognition. Por ejemplo, si inicia demasiados trabajos de Amazon Rekognition Vídeo al mismo tiempo, las llamadas para iniciar operaciones, como `StartLabelDetection`, generan una excepción `LimitExceededException` (código de estado HTTP: 400) hasta que el número de trabajos que se ejecutan al mismo tiempo esté por debajo del límite de servicio de Amazon Rekognition.

¿Reintentar? No

ProvisionedThroughputExceededException

Mensajes:

- Se ha superado la tasa aprovisionada.
- Se ha superado el límite de descarga de S3.

El número de solicitudes ha superado su límite de rendimiento. Para obtener más información, consulte Límites de servicio de Amazon Rekognition.

Para solicitar un aumento de los límites, siga las instrucciones de the section called “Crear un caso para cambiar las cuotas de TPS”.

¿Reintentar? Sí

ResourceAlreadyExistsException

Mensaje: El ID de colección <ID de colección> ya existe.

Ya existe una colección con el ID especificado.

¿Reintentar? No

ResourceInUseException

Mensajes:

- Nombre del procesador de flujo ya en uso.
- El recurso especificado está en uso.
- El procesador no está disponible para detener el flujo.
- No se puede eliminar el procesador de flujos.

Vuelva a intentarlo cuando el recurso esté disponible.

¿Reintentar? No

ResourceNotFoundException

Mensaje: Varios mensajes en función de la llamada a la API.

El recurso especificado no existe.

¿Reintentar? No

ThrottlingException

Mensaje: Vaya más despacio; aumento repentino de la tasa de solicitudes.

Su tasa de aumento de solicitudes es demasiado rápida. Reduzca la tasa de solicitudes y auméntela gradualmente. Le recomendamos que interrumpa exponencialmente y vuelva a intentarlo. De forma predeterminada, AWS SDKs utilizan la lógica de reintento automático y el retroceso exponencial. Para obtener más información, consulte Reintentos de error y retardo exponencial en AWS y Retardo exponencial y fluctuación.

¿Reintentar? Sí

VideoTooLargeException

Mensaje: El tamaño del vídeo en bytes: <Tamaño de vídeo> es mayor que el límite máximo de: <Tamaño máximo> bytes.

El tamaño del archivo o la duración del medio suministrado es demasiado grande. Para obtener más información, consulte Directrices y cuotas en Amazon Rekognition.

¿Reintentar? No

Código de estado HTTP 5xx

Un código de estado HTTP `5xx` indica un problema cuya resolución corresponde a AWS. Posiblemente sea un error temporal. Si lo es, puede volver a intentar su solicitud hasta que se ejecute satisfactoriamente. En caso contrario, vaya al Panel de estado del servicio de AWS para comprobar si existe algún problema funcional con el servicio.

InternalServerError (HTTP 500)

Mensaje: Error de servidor interno

Amazon Lex ha tenido un problema de servicio. Pruebe la llamada de nuevo. Debe efectuar una interrupción exponencial y volver a intentarlo. De forma predeterminada, AWS SDKs utilizan la lógica de reintento automático y el retroceso exponencial. Para obtener más información, consulte Reintentos de error y retardo exponencial en AWS y Retardo exponencial y fluctuación.

¿Reintentar? Sí

ThrottlingException (HTTP 500)

Mensaje: Servicio no disponible

Amazon Lex no puede procesar temporalmente la solicitud. Pruebe la llamada de nuevo. Le recomendamos que interrumpa exponencialmente y vuelva a intentarlo. De forma predeterminada, AWS SDKs utilizan la lógica de reintento automático y el retroceso exponencial. Para obtener más información, consulte Reintentos de error y retardo exponencial en AWS y Retardo exponencial y fluctuación.

¿Reintentar? Sí

Control de errores en la aplicación

Para que la aplicación funcione sin problemas, debe añadir lógica que capture los errores y responda a ellos. Los enfoques habituales incluyen el uso de bloques `try-catch` o instrucciones `if-then`.

La AWS SDKs realiza sus propios reintentos y comprobaciones de errores. Si se produce un error al utilizar uno de los sistemas de AWS SDKs, el código y la descripción del error pueden ayudarle a solucionarlo.

También debería aparecer el `Request ID` en la respuesta. El `Request ID` puede resultar útil si tiene que acudir a AWS Support para diagnosticar el problema.

En el siguiente fragmento de código Java se intentan detectar objetos en una imagen y se realiza un control de errores rudimentario. En este caso, informa al usuario de que se ha producido un error en la solicitud.

```
try {
    DetectLabelsResult result = rekognitionClient.detectLabels(request);
    List <Label> labels = result.getLabels();

    System.out.println("Detected labels for " + photo);
    for (Label label: labels) {
        System.out.println(label.getName() + ": " + label.getConfidence().toString());
    }
}
catch(AmazonRekognitionException e) {
    System.err.println("Could not complete operation");
    System.err.println("Error Message:  " + e.getMessage());
    System.err.println("HTTP Status:    " + e.getStatusCode());
    System.err.println("AWS Error Code: " + e.getErrorCode());
    System.err.println("Error Type:     " + e.getErrorType());
    System.err.println("Request ID:     " + e.getRequestId());
}
catch (AmazonClientException ace) {
    System.err.println("Internal error occurred communicating with Rekognition");
    System.out.println("Error Message:  " + ace.getMessage());
}
```

En este fragmento de código, la construcción `try-catch` controla dos tipos de excepciones diferentes:

- `AmazonRekognitionException`: esta excepción se produce si la solicitud del cliente se ha transmitido correctamente a Amazon Rekognition, pero éste no ha podido procesarla y ha devuelto una respuesta de error.
- `AmazonClientException`: esta excepción se genera si el cliente no ha podido obtener una respuesta de un servicio o sí la ha obtenido pero no ha podido analizarla.

Utilizar Amazon Rekognition como servicio autorizado de FedRAMP

El programa de conformidad con FedRAMP de AWS incluye a Amazon Rekognition como servicio autorizado de FedRAMP. Si es un cliente comercial o de una administración federal, puede utilizar el servicio para procesar y almacenar cargas de trabajo confidenciales en las regiones EE. UU. Este y EE. UU. Oeste de AWS, para datos hasta el nivel de impacto moderado. Puede utilizar el servicio para cargas de trabajo confidenciales en el límite de autorización de la región AWS GovCloud (EE. UU.) para datos hasta el nivel de impacto alto. Para obtener más información acerca de la conformidad con FedRAMP, consulte Cumplimiento con FedRAMP de AWS.

Para ser compatible con FedRAMP, puede utilizar un extremo del punto de conexión Estándar Federal de Procesamiento de Información (FIPS). Esto le da acceso a módulos criptográficos validados por FIPS 140-2 cuando trabaja con información confidencial. Para obtener más información sobre los puntos de conexión de FIPS, consulte Información general sobre FIPS 140-2.

Puede utilizar AWS Command Line Interface (AWS CLI) o uno de los SDK de AWS para especificar el punto de conexión utilizado por Amazon Rekognition.

Para obtener información sobre los puntos de conexión que se pueden utilizar con Amazon Rekognition, consulte Regiones y puntos de conexión de Amazon Rekognition.

A continuación se muestran ejemplos del tema Colecciones de descripciones de la Guía para desarrolladores de Amazon Rekognition. Se modifican para especificar el punto de conexión de Región y FIPS mediante el cual se accede a Amazon Rekognition.

Java

Para Java, utilice el método `withEndpointConfiguration` cuando construya el cliente de Amazon Rekognition. Este ejemplo muestra las colecciones que tiene que utilizar el punto de conexión de FIPS en la región de EE. UU. Este (Norte de Virginia):

```
//Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
//PDX-License-Identifier: MIT-0 (For details, see https://github.com/awsdocs/amazon-
rekognition-developer-guide/blob/master/LICENSE-SAMPLECODE.)

package aws.example.rekognition.image;

import java.util.List;

import com.amazonaws.services.rekognition.AmazonRekognition;
import com.amazonaws.services.rekognition.AmazonRekognitionClientBuilder;
import com.amazonaws.services.rekognition.model.ListCollectionsRequest;
import com.amazonaws.services.rekognition.model.ListCollectionsResult;

public class ListCollections {

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


      AmazonRekognition amazonRekognition =
 AmazonRekognitionClientBuilder.standard()
         .withEndpointConfiguration(new
 AwsClientBuilder.EndpointConfiguration("https://rekognition-fips.us-
east-1.amazonaws.com","us-east-1"))
         .build();


      System.out.println("Listing collections");
      int limit = 10;
      ListCollectionsResult listCollectionsResult = null;
      String paginationToken = null;
      do {
         if (listCollectionsResult != null) {
            paginationToken = listCollectionsResult.getNextToken();
         }
         ListCollectionsRequest listCollectionsRequest = new
 ListCollectionsRequest()
                 .withMaxResults(limit)
                 .withNextToken(paginationToken);

 listCollectionsResult=amazonRekognition.listCollections(listCollectionsRequest);

         List < String > collectionIds = listCollectionsResult.getCollectionIds();
         for (String resultId: collectionIds) {
```

```
            System.out.println(resultId);
         }
      } while (listCollectionsResult != null &&
 listCollectionsResult.getNextToken() !=
         null);

   }
}
```

AWS CLI

Para AWS CLI, utilice el argumento `--endpoint-url` para especificar el punto de conexión a través del cual se accede a Amazon Rekognition. Este ejemplo muestra las colecciones que tiene que utilizar el punto de conexión de FIPS en la región Este de EE. UU. (Ohio):

```
aws rekognition list-collections --endpoint-url https://rekognition-fips.us-
east-2.amazonaws.com --region us-east-2
```

Python

Para Python, use el argumento `endpoint_url` en la función boto3.client. Establézcalo en el punto de conexión que quiera especificar. Este ejemplo muestra las colecciones que tiene que utilizar el punto de conexión de FIPS en la región de Oeste de EE. UU. (Oregón):

```
#Copyright 2018 Amazon.com, Inc. or its affiliates. All Rights Reserved.
#PDX-License-Identifier: MIT-0 (For details, see https://github.com/awsdocs/amazon-
rekognition-developer-guide/blob/master/LICENSE-SAMPLECODE.)

import boto3

def list_collections():

    max_results=2

    client=boto3.client('rekognition', endpoint_url='https://rekognition-fips.us-
west-2.amazonaws.com', region_name='us-west-2')

    #Display all the collections
    print('Displaying collections...')
    response=client.list_collections(MaxResults=max_results)
    collection_count=0
    done=False
```

```
    while done==False:
        collections=response['CollectionIds']

        for collection in collections:
            print (collection)
            collection_count+=1
        if 'NextToken' in response:
            nextToken=response['NextToken']

 response=client.list_collections(NextToken=nextToken,MaxResults=max_results)

        else:
            done=True

    return collection_count

def main():

    collection_count=list_collections()
    print("collections: " + str(collection_count))
if __name__ == "__main__":
    main()
```

Prácticas recomendadas para sensores, vídeos e imágenes de entrada

Esta sección contiene información acerca de las prácticas recomendadas para utilizar Amazon Rekognition. Estas prácticas recomendadas le ayudarán a obtener un rendimiento óptimo de las operaciones que invoque. Si tiene dificultades para obtener los resultados esperados de una operación, asegúrese de seguir las prácticas recomendadas aquí documentadas.

Para obtener información sobre la latencia de las operaciones de imágenes, consulte lo siguiente:

- the section called “Latencia de operación de Amazon Rekognition Image”

Las operaciones de comparación facial y búsqueda de rostros requieren que siga prácticas recomendadas específicas para encontrar rostros en una imagen. Estos requisitos también se han documentado en el enlace siguiente:

- the section called “Recomendaciones para la comparación de rostros en las imágenes de entrada”
- the section called “Recomendaciones para buscar rostros en una colección”

En las siguientes secciones se describe cómo configurar la cámara para cada tipo de contenido multimedia que Amazon Rekognition es capaz de analizar:

- the section called “Recomendaciones de configuración de la cámara (imagen y vídeo)”
- the section called “Recomendaciones de configuración de la cámara (vídeo almacenado y en streaming)”
- the section called “Recomendaciones de configuración de la cámara (vídeo en streaming)”

Las operaciones de Face Liveness también tienen sus prácticas recomendadas propias, que deben seguirse para obtener el mejor rendimiento de la herramienta de comprobación de Liveness:

- the section called “Recomendaciones para el uso de Face Liveness”

Latencia de operación de Amazon Rekognition Image

Para garantizar la menor latencia posible para las operaciones de Amazon Rekognition Image tenga en cuenta lo siguiente:

- La región del bucket de Amazon S3 que contiene las imágenes debe coincidir con la región que utiliza para las operaciones de API de Amazon Rekognition Image.
- Llamar a una operación de Amazon Rekognition Image con bytes de imagen es más rápido que cargar la imagen en un bucket de Amazon S3 y, a continuación, hacer referencia a la imagen subida en una operación de Amazon Rekognition Image. Tenga en cuenta este enfoque si está cargando imágenes en Amazon Rekognition Image para procesamiento casi en tiempo real. Por ejemplo, las imágenes cargadas desde una cámara IP o las imágenes cargadas a través de un portal web.
- Si la imagen ya está en un bucket de Amazon S3, probablemente sea más rápido hacer referencia a ella en una operación de Amazon Rekognition Image que transferir bytes de imagen a la operación.

Recomendaciones para la comparación de rostros en las imágenes de entrada

Los modelos que se utilizan en las operaciones de reconocimiento de rostros están diseñados para funcionar con una amplia variedad de posturas, expresiones faciales, rangos de edad, rotaciones, condiciones de iluminación y tamaños. Le recomendamos que aplique las siguientes directrices cuando elija las fotos de referencia de CompareFaces o si desea agregar rostros a una colección utilizando IndexFaces.

Recomendaciones generales sobre la introducción de imágenes para operaciones faciales

- Utilice imágenes que sean brillantes y nítidas. En la medida de lo posible, evite utilizar imágenes que aparezcan borrosas debido al movimiento de la cámara o del sujeto. Puede utilizar DetectFaces para determinar el brillo y la nitidez de un rostro.
- Para detectar la mirada, se recomienda subir la imagen original con el tamaño y la calidad originales.

- Utilice una imagen con un rostro comprendido en el rango recomendado de ángulos. El ángulo de rotación sobre el eje X debe ser inferior a 30 grados hacia abajo y a 45 grados hacia arriba. El ángulo de rotación sobre el eje Y debe ser inferior a 45 grados en ambos sentidos. No hay ninguna restricción en la rotación sobre el eje Z.
- Utilice una imagen de un rostro con ambos ojos abiertos y visibles.
- Utilice una imagen del rostro que no esté oscurecida ni demasiado recortada. La imagen debe incluir toda la cabeza y los hombros de la persona. No debe recortarse para el cuadro delimitador del rostro.
- Evite los elementos que enmascaran el rostro, como diademas o máscaras.
- Utilice una imagen de un rostro que ocupe una gran proporción de la imagen. Se hallan coincidencias más precisas de las imágenes en las que el rostro ocupa una proporción mayor del espacio total.
- Asegúrese de que las imágenes sean lo suficientemente grandes en cuanto a resolución. Amazon Rekognition puede reconocer rostros tan pequeños de hasta 50 x 50 píxeles en resoluciones de imagen de hasta 1920 x 1080. Las imágenes de resolución más alta requieren un tamaño de rostro mínimo mayor. Los rostros que tienen un tamaño superior al mínimo ofrecen un conjunto de resultados más preciso en la comparación facial.
- Utilice imágenes de color.
- Utilice imágenes con una iluminación plana del rostro, es decir, cuya iluminación no produzca sombras.
- Utilice imágenes que tengan un contraste suficiente respecto al fondo. Un fondo monocromo de alto contraste funciona bien.
- Para las aplicaciones que requieran un alto nivel de precisión, utilice imágenes de rostros con expresiones faciales neutras, la boca cerrada y sin sonrisa.

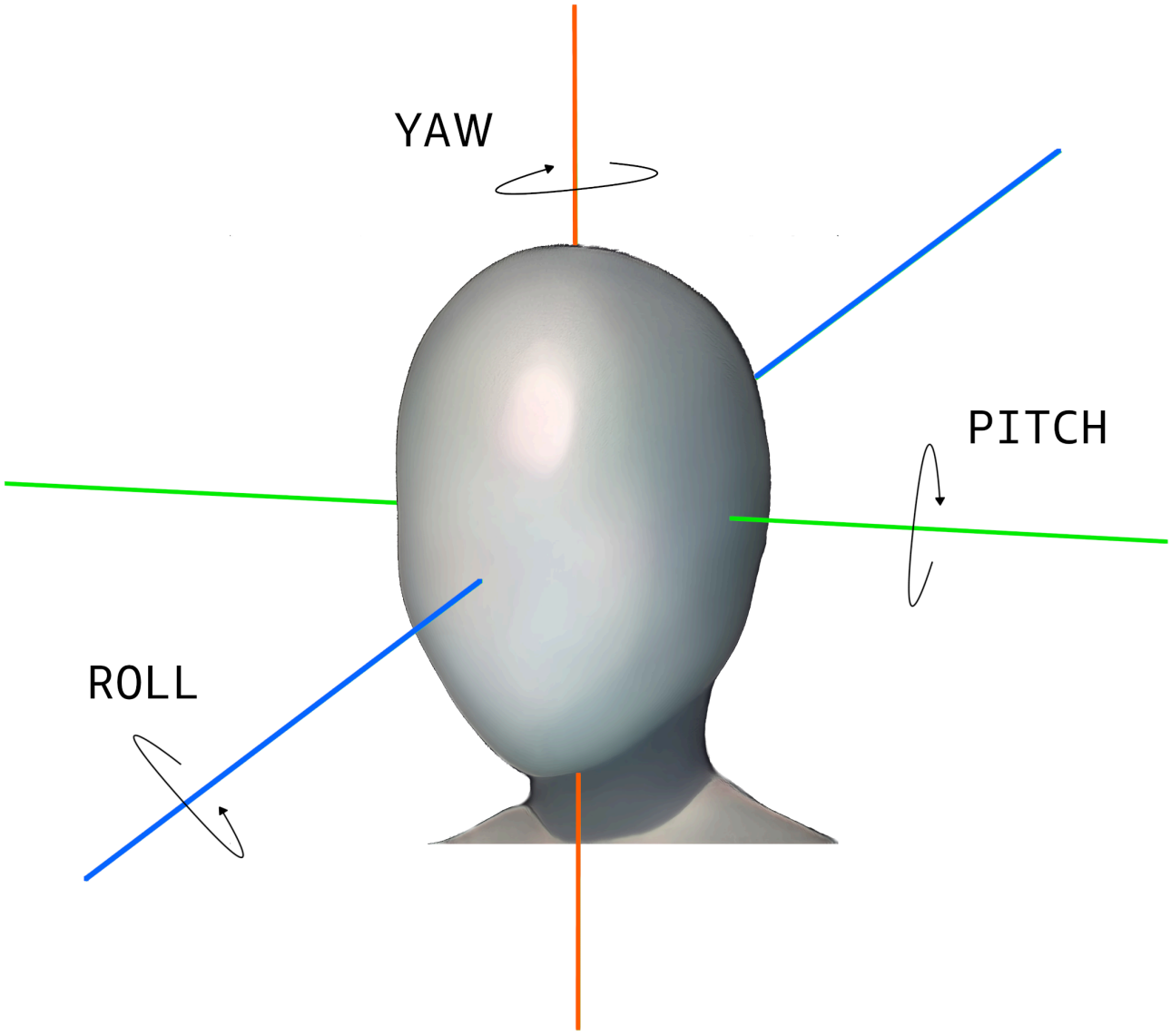
Recomendaciones para buscar rostros en una colección

- Cuando busque rostros en una colección, asegúrese de que las imágenes de rostros recientes estén indexadas.
- Al crear una colección mediante `IndexFaces`, utilice varias imágenes del rostro de una persona con diferentes ángulos de giro sobre los ejes X e Y (dentro del rango recomendado de ángulos). Recomendamos que disponer de al menos 5 imágenes del rostro de la persona que se va a indexar: mirando directamente a cámara, girado hacia la izquierda con un índice de rotación sobre el eje Y de 45 grados o menos, girado hacia la derecha con un índice de rotación sobre el eje Y de

45 grados o menos, inclinado hacia abajo con un índice de rotación sobre el eje X de 30 grados o menos e inclinado hacia arriba con un índice de rotación sobre el eje X de 45 grados o menos. Si desea realizar el seguimiento del hecho de que estas instancias del rostro pertenecen a la misma persona, puede ser conveniente utilizar el atributo externo de ID de imagen si la imagen indexada contiene un solo rostro. Por ejemplo, se pueden rastrear cinco imágenes de John Doe en la colección con una imagen externa IDs como`John_Doe_1.jpg, … John_Doe_5.jpg`.

Recomendaciones de configuración de la cámara (imagen y vídeo)

Las siguientes recomendaciones son adicionales a las que se indican en Recomendaciones para la comparación de rostros en las imágenes de entrada.



- Resolución de imagen: No hay ningún requisito mínimo de resolución de imagen, siempre y cuando la resolución del rostro sea de 50 x 50 píxeles en imágenes con una resolución total de hasta 1920 x 1080. Las imágenes de resolución más alta requieren un tamaño de rostro mínimo mayor.

> **Note**
>
> La recomendación anterior se basa en la resolución nativa de la cámara. Generar una imagen de alta resolución a partir de una imagen de baja resolución no produce los resultados necesarios para la búsqueda de rostros (debido a los artefactos generados por el muestreo ascendente de la imagen).

- Ángulo de la cámara: existen tres mediciones del ángulo de la cámara: la rotación sobre el eje X, la rotación sobre el eje Z y la rotación sobre el eje Y.
  - Rotación sobre el eje X: recomendamos que sea inferior a 30 grados cuando la cámara mira hacia abajo e inferior a 45 grados cuando la cámara mira hacia arriba.
  - Rollo: no hay un requisito mínimo para este parámetro. Amazon Rekognition puede procesar cualquier cantidad de rollos.
  - Guiñada: recomendamos una guiñada de menos de 45 grados en cualquier dirección.

  El ángulo de rotación del rostro sobre cualquiera de los ejes que la cámara captura es una combinación del ángulo de la cámara respecto a la escena y el ángulo en que se encuentra el rostro del sujeto en esa escena. Por ejemplo, si la cámara está inclinada 30 grados hacia abajo y la persona tiene la cabeza inclinada 30 grados, el ángulo real de rotación del rostro sobre el eje X que observa la cámara será de 60 grados. En este caso, Amazon Rekognition no podrá reconocer el rostro. Recomendamos configurar las cámaras de tal forma que sus ángulos se basen en el supuesto de que lo habitual es que las personas miren a la cámara con un ángulo de rotación sobre el eje X (combinado del rostro y de la cámara) de 30 grados o menos.
- Zoom de la cámara: Este parámetro de la cámara debe definirse en función de la resolución mínima recomendada del rostro de 50 x 50 píxeles. Recomendamos utilizar la configuración de zoom de la cámara de tal forma que los rostros deseados presenten una resolución que no sea inferior a 50 x 50 píxeles.
- Altura de la cámara: este parámetro debe basarse en el ángulo de rotación sobre el eje X de la cámara.

Recomendaciones de configuración de la cámara (vídeo almacenado y en streaming)

Las siguientes recomendaciones son adicionales a las que se indican en Recomendaciones de configuración de la cámara (imagen y vídeo).

- El códec debe estar codificado en formato H.264.
- La velocidad de fotogramas recomendada es de 30 fps. No debe ser menor que 5 fps.
- La velocidad de bits recomendada del codificador es de 3 Mbps. No debe ser menor que 1,5 Mbps.
- Resolución de fotogramas frente a velocidad de fotogramas: si la velocidad de bits del codificador está restringida, recomendamos dar prioridad a la mayor resolución de fotogramas respecto a la velocidad de fotogramas, para obtener mejores resultados de búsqueda de rostros. De este modo, Amazon Rekognition obtendrá el fotograma de mayor calidad dentro de la velocidad de bits asignada. Sin embargo, esto tiene un inconveniente. Debido a la baja velocidad de fotogramas, la cámara pierde los movimientos rápidos de una escena. Es importante entender las contrapartidas que cada uno de estos dos parámetros tiene en una configuración determinada. Por ejemplo, si la velocidad de bits máxima posible es de 1,5 Mbps, una cámara puede capturar 1080p a 5 fps o 720p a 15 fps. La elección de una de estas dos opciones depende de la aplicación, siempre y cuando se cumpla la resolución recomendada del rostro de 50 x 50 píxeles.

Recomendaciones de configuración de la cámara (vídeo en streaming)

La siguiente recomendación es adicional a las que se indican en Recomendaciones de configuración de la cámara (vídeo almacenado y en streaming).

Una restricción adicional en las aplicaciones de streaming es el ancho de banda de Internet. Para el vídeo en directo, Amazon Rekognition solo acepta Amazon Kinesis Video Streams como elemento de entrada. Es importante comprender la dependencia entre la velocidad de bits del codificador y el ancho de banda disponible de la red. Como mínimo, el ancho de banda disponible debe admitir la misma velocidad de bits que utilice la cámara para codificar la transmisión en directo. Esto garantiza que aquello que capture la cámara se transmita a través de Amazon Kinesis Video Streams. Si el ancho de banda disponible es inferior a la velocidad de bits del codificador, Amazon Kinesis Video Streams perderá bits en función del ancho de banda de la red. Como consecuencia, la calidad del vídeo será baja.

En una configuración de streaming típica, se conectan varias cámaras a un hub de red que retransmite las secuencias. En este caso, el ancho de banda debería dar cabida a la suma acumulativa de las secuencias procedentes de todas las cámaras conectadas al hub. Por ejemplo, si el hub está conectado a cinco cámaras que codifican a 1,5 Mbps, el ancho de banda disponible en la red debería ser de al menos 7,5 Mbps. Para asegurarse de que no se pierdan paquetes, es conveniente que el ancho de banda de la red sea superior a 7,5 Mbps, con el fin de admitir las interferencias debidas a las interrupciones en la conexión entre la cámara y el hub. El valor real dependerá de la fiabilidad de la red interna.

Recomendaciones para el uso de Face Liveness

Es conveniente que siga estas prácticas recomendadas si va a utilizar Rekognition Face Liveness:

- Los usuarios deben comprobar la vitalidad del rostro en entornos que no sean demasiado oscuros ni demasiado brillantes y que tengan una iluminación bastante uniforme.
- Los usuarios deben aumentar el brillo de la pantalla hasta su nivel máximo al realizar comprobaciones en los navegadores web. Mobile Native SDKs ajusta el brillo de la pantalla automáticamente.
- Elija un umbral de puntuación de confianza que refleje la naturaleza de su caso de uso. Para los casos de uso con mayores problemas de seguridad, use un umbral alto.
- Realice controles humanos periódicos de las imágenes de auditoría para asegurarse de que los ataques simulados se mitiguen al alcanzar el umbral de confianza que haya establecido.
- Ofrezca una ruta alternativa de verificación de la vitalidad del rostro a sus usuarios si son fotosensibles o no quieren verificar la vitalidad del rostro mediante Rekognition.
- No envíe ni muestre la puntuación del control de vitalidad en la aplicación de usuario. Envíe únicamente una señal de aprobación o rechazo.
- Permita solo cinco comprobaciones de funcionamiento fallidas en tres minutos desde un solo dispositivo. Si se producen cinco errores, se deja que el usuario espere entre 30 y 60 minutos. Si el patrón se repite de 3 a 5 veces, bloquee el dispositivo del usuario para que no pueda realizar llamadas adicionales.
- Implemente la pantalla de preparación en su flujo de trabajo para que los usuarios puedan pasar más fácilmente las comprobaciones de vitalidad del rostro.
- Usted es responsable de proporcionar avisos de privacidad legalmente adecuados a sus usuarios finales y de obtener el consentimiento necesario de ellos para el procesamiento, almacenamiento, uso y transferencia de contenido por parte de Face Liveness.

Detección de objetos y conceptos

Esta sección proporciona información para detectar etiquetas en imágenes y vídeos con Amazon Rekognition Image y Amazon Rekognition Video.

Una etiqueta o marca es un objeto o concepto (incluidos escenas y acciones) encontrado en una imagen o vídeo basado en su contenido. Por ejemplo, una imagen de personas en una playa tropical podría contener etiquetas como Palmera (objetos), Playa (escena), Correr (acción) y Exterior (concepto).

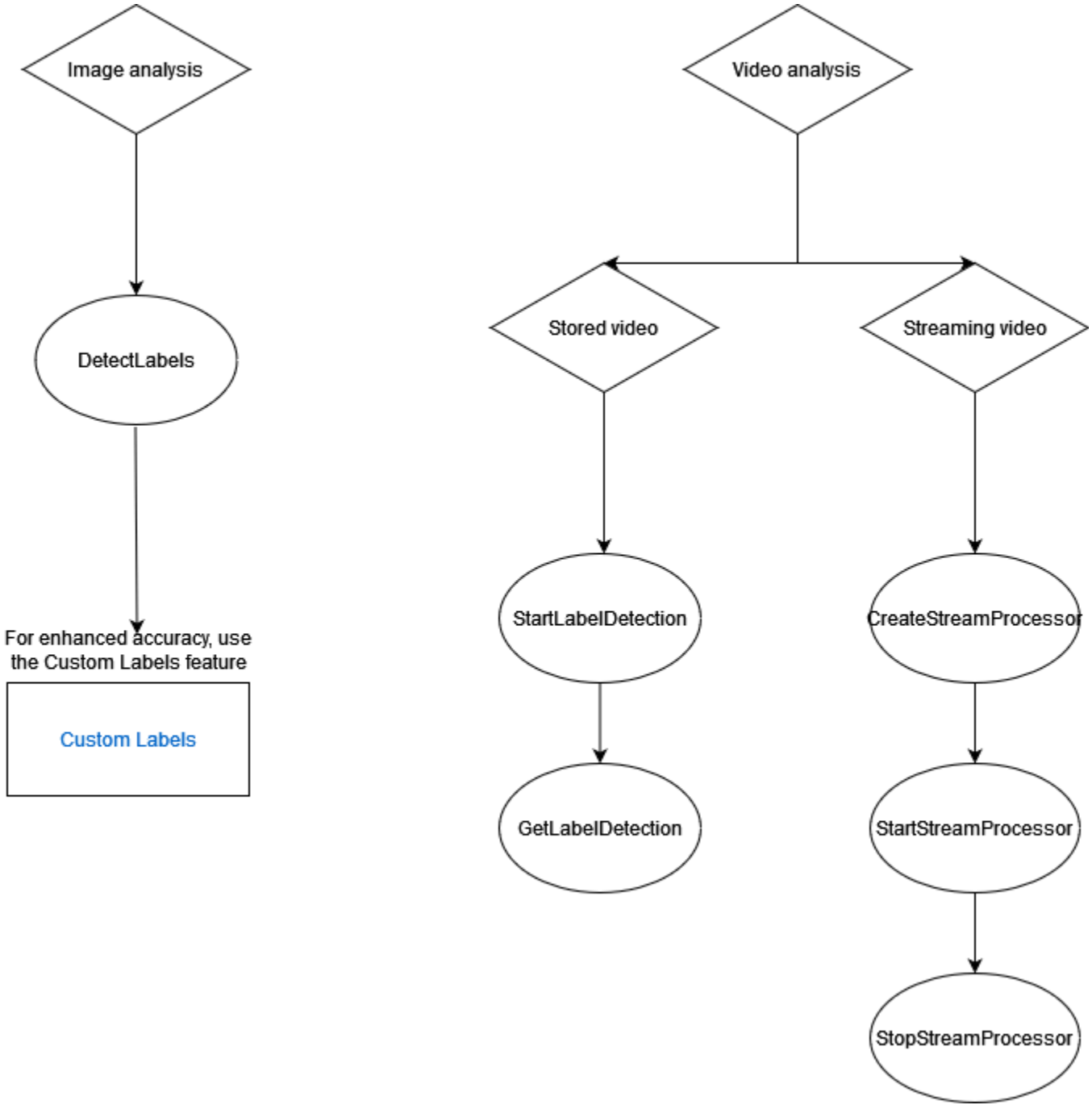
Etiquetas compatibles con las operaciones de detección de etiquetas de Rekognition

- Para descargar la lista más reciente de etiquetas y cuadros delimitadores de objetos compatibles con Amazon Rekognition, haga clic aquí.
- Para descargar la lista anterior de etiquetas y cuadros delimitadores de objetos, haga clic aquí.

> **Note**
>
> Amazon Rekognition realiza predicciones binarias de género (hombre, mujer, niña, etc.) en función de la apariencia física de una persona en una imagen determinada. Este tipo de predicción no está diseñada para categorizar la identidad de género de una persona, y no debería usar Amazon Rekognition para tomar esa determinación. Por ejemplo, se podría predecir que un actor masculino que lleva una peluca de pelo largo y pendientes para un papel es una mujer.
>
> El uso de Amazon Rekognition para realizar predicciones binarias de género es el más adecuado para los casos de uso en los que es necesario analizar las estadísticas agregadas de distribución de género sin identificar a usuarios específicos. Por ejemplo, el porcentaje de usuarios que son mujeres en comparación con los hombres en una plataforma de redes sociales.
>
> No es recomendable utilizar predicciones binarias de género para tomar decisiones que podrían afectar a los derechos, la privacidad o el acceso de una persona a los servicios.

Amazon Rekognition devuelve las etiquetas en inglés. Puede usar Amazon Translate para traducir etiquetas del inglés a otros idiomas.

En el siguiente diagrama se muestra el orden de las operaciones de llamada, en función de sus objetivos de uso de las operaciones de Amazon Rekognition Image o Amazon Rekognition Video:



Etiquetado de objetos de respuesta

Cuadro delimitador

Amazon Rekognition Image y Amazon Rekognition Video pueden devolver el cuadro delimitador de etiquetas de objetos comunes, como personas, automóviles, muebles, prendas de vestir o mascotas. La información del cuadro delimitador no se devuelve en el caso de las etiquetas de objetos menos comunes. Puede utilizar los cuadros delimitadores para encontrar las ubicaciones exactas de objetos en una imagen, contar cuántas veces aparece el objeto detectado o medir el tamaño de un objeto mediante las dimensiones del cuadro delimitador.

Por ejemplo, en la imagen siguiente, Amazon Rekognition Image es capaz de detectar la presencia de una persona, un patinete, coches aparcados y otra información. Amazon Rekognition Image también devuelve el recuadro delimitador de una persona detectada y otros objetos detectados, como coches y ruedas.



Puntuación de confianza

Amazon Rekognition Video y Amazon Rekognition Image proporcionan además una puntuación de porcentaje sobre la confianza que tiene Amazon Rekognition en la precisión de cada etiqueta detectada.

Elementos principales

Amazon Rekognition Image y Amazon Rekognition Video utilizan una taxonomía jerárquica de etiquetas antecesoras para categorizar las etiquetas. Por ejemplo, una persona que está cruzando a pie la calle podría detectarse como Pedestrian (Peatón). La etiqueta principal de Pedestrian (Peatón) es Person (Persona). Ambas etiquetas se devuelven en la respuesta. Se devuelven todas las etiquetas antecesoras. Además, una etiqueta determinada contiene una lista de su etiqueta principal y demás etiquetas antecesoras. Por ejemplo, las etiquetas "abuelas" y "bisabuelas", si las hay. Puede utilizar etiquetas principales para crear grupos de etiquetas relacionadas y hacer posibles las consultas de etiquetas similares en una o varias imágenes. Por ejemplo, una consulta de todas las etiquetas Vehicle (Vehículo) podría devolver un automóvil de una imagen y una motocicleta de otra.

Categorías

Amazon Rekognition Image y Amazon Rekognition Video devuelven información en las categorías de etiquetas. Las etiquetas forman parte de categorías que agrupan etiquetas individuales en función de funciones y contextos comunes, como «Vehículos y automoción» y «Alimentos y bebidas». Una categoría de etiquetas puede ser una subcategoría de una categoría principal.

Alias

Además de devolver las etiquetas, Amazon Rekognition Image y Amazon Rekognition Video devuelven cualquier alias asociado a la etiqueta. Los alias son etiquetas con el mismo significado o etiquetas que se pueden intercambiar visualmente con la etiqueta principal devuelta. Por ejemplo, «Móvil» es un alias de «Teléfono móvil».

En versiones anteriores, Amazon Rekognition Image mostraba alias como «Móvil» en la misma lista de nombres de etiquetas principales que contenían «Teléfono móvil». Amazon Rekognition Image ahora muestra «Móvil» en un campo denominado «alias» y «Teléfono móvil» en la lista de nombres de etiquetas principales. Si su aplicación se basa en las estructuras devueltas por una versión anterior de Rekognition, es posible que necesite transformar la respuesta actual devuelta por

las operaciones de detección de etiquetas de imagen o vídeo en la estructura de respuesta anterior, en la que todas las etiquetas y alias se devuelven como etiquetas principales.

Si necesita transformar la respuesta actual de la DetectLabels API (para la detección de etiquetas en imágenes) en la estructura de respuesta anterior, consulte el ejemplo de código enTransformar la DetectLabels respuesta.

Si necesita transformar la respuesta actual de la GetLabelDetection API (para la detección de etiquetas en los vídeos almacenados) en la estructura de respuesta anterior, consulte el ejemplo de código enTransformando la GetLabelDetection respuesta.

Propiedades de la imagen

Amazon Rekognition Image devuelve información sobre la calidad de la imagen (nitidez, brillo y contraste) de toda la imagen. La nitidez y el brillo también se devuelven para el primer plano y el fondo de la imagen. Las propiedades de la imagen también se pueden utilizar para detectar los colores dominantes de toda la imagen, el primer plano, el fondo y los objetos con cuadros delimitadores.



A continuación, se muestra un ejemplo de los ImageProperties datos contenidos en la respuesta de una DetectLabels operación para la imagen en curso:

Image Properties	Dominant Colors Examples and Pixel Percentage	Image Quality
Entire Image	Hex code #808080, RGB (128, 128, 128), 15.72 Hex code #000000, RGB (0, 0, 0), 15.10 Hex code #696969, RGB (105, 105, 105), 14.02 Hex code #8fbc8f, RGB (143, 188, 143), 12.70 Hex code #5f9ea0, RGB (95, 158, 160), 11.92	Brightness: 76.08 Sharpness: 89.72 Contrast: 88.42
Foreground	Hex code #8fbc8f, RGB (143, 188, 143), 30.18 Hex code #5f9ea0, RGB (95, 158, 160), 24.29 Hex code #000000, RGB (0, 0, 0), 12.02 Hex code #2f4f4f, RGB (47, 79, 79), 9.20 Hex code #696969, RGB (105, 105, 105), 8.95	Brightness: 79.48 Sharpness: 93.47
Background	Hex code #808080, RGB (128, 128, 128), 21.16 Hex code #2f4f4f, RGB (47, 79, 79), 14.61 Hex code #000000, RGB (0, 0, 0), 14.23 Hex code #696969, RGB (105, 105, 105), 13.19 Hex code #ffebcd, RGB (255, 235, 205), 12.80	Brightness: 74.42 Sharpness: 87.84
Car (example of objects with bounding boxes)	Hex code #5f9ea0, RGB (95, 158, 160), 29.18 Hex code #8fbc8f, RGB (143, 188, 143), 14.39 Hex code #000000, RGB (0, 0, 0), 11.76 Hex code #808080, RGB (128, 128, 128), 11.38 Hex code #2f4f4f, RGB (47, 79, 79), 9.44	Not applicable

Las propiedades de imagen no están disponibles para Amazon Rekognition Video.

Versión del modelo

Amazon Rekognition Image y Amazon Rekognition Video devuelven la versión del modelo de detección de etiquetas que se ha utilizado para detectar etiquetas en una imagen o un vídeo almacenado.

Filtros de inclusión y exclusión

Puede filtrar los resultados devueltos por las operaciones de detección de etiquetas de Amazon Rekognition Image y Amazon Rekognition Video. Filtre los resultados proporcionando criterios de filtrado para las etiquetas y las categorías. Los filtros de etiquetas pueden ser inclusivos o exclusivos.

Consulte Detección de etiquetas en una imagen para obtener más información sobre el filtrado de los resultados obtenidos con `DetectLabels`.

Consulte Detección de etiquetas en un vídeo para obtener más información sobre el filtrado de los resultados obtenidos por `GetLabelDetection`.

Clasificación y agregación de los resultados

Los resultados obtenidos de determinadas operaciones de Amazon Rekognition Video se pueden ordenar y agregar según las marcas de tiempo y los segmentos de vídeo. Al recuperar los resultados de un trabajo de detección de etiquetas o moderación de contenido, con `GetLabelDetection` o `GetContentModeration` respectivamente, puede utilizar los argumentos `SortBy` y `AggregateBy` para especificar cómo desea que se devuelvan los resultados. Puede usar `SortBy` con `TIMESTAMP` o `NAME` (nombres de etiqueta) y usar `TIMESTAMPS` o `SEGMENTS` con el AggregateBy argumento.

Detección de etiquetas en una imagen

Puede utilizar la DetectLabelsoperación para detectar etiquetas (objetos y conceptos) en una imagen y recuperar información sobre las propiedades de la imagen. Las propiedades de la imagen incluyen atributos como el color del primer plano y del fondo y la nitidez, el brillo y el contraste de la imagen. Puede recuperar solo las etiquetas de una imagen, solo las propiedades de la imagen o ambas. Para ver un ejemplo, consulta Análisis de imágenes almacenadas en un bucket de Amazon S3.

En los ejemplos siguientes se utilizan varios AWS SDKs y los AWS CLI para llamar`DetectLabels`. Para obtener más información sobre la respuesta de la operación `DetectLabels`, consulte DetectLabels respuesta.

Para detectar las etiquetas en una imagen

1. Si aún no lo ha hecho:
   a. Cree o actualice un usuario con los permisos `AmazonRekognitionFullAccess` y `AmazonS3ReadOnlyAccess`. Para obtener más información, consulte Paso 1: Configurar una cuenta de AWS y crear un usuario.
   b. Instale y configure el AWS CLI y el AWS SDKs. Para obtener más información, consulte Paso 2: Configure y AWS CLI AWS SDKs.
2. Suba una imagen que contenga uno o varios objetos, como árboles, casas o barcos, en el bucket de S3. La imagen debe estar en formato .jpg o .png.

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

3. Consulte los siguientes ejemplos para llamar a la operación `DetectLabels`.

   Java

   Este ejemplo muestra una lista de las etiquetas que se han detectado en la imagen de entrada. Reemplace los valores de `bucket` y `photo` por los nombre del bucket de Amazon S3 y de la imagen que utilizó en el paso 2.

```
package com.amazonaws.samples;
import java.util.List;

import com.amazonaws.services.rekognition.model.BoundingBox;
import com.amazonaws.services.rekognition.model.DetectLabelsRequest;
import com.amazonaws.services.rekognition.model.DetectLabelsResult;
import com.amazonaws.services.rekognition.model.Image;
import com.amazonaws.services.rekognition.model.Instance;
import com.amazonaws.services.rekognition.model.Label;
import com.amazonaws.services.rekognition.model.Parent;
import com.amazonaws.services.rekognition.model.S3Object;
import com.amazonaws.services.rekognition.AmazonRekognition;
import com.amazonaws.services.rekognition.AmazonRekognitionClientBuilder;
import com.amazonaws.services.rekognition.model.AmazonRekognitionException;

public class DetectLabels {

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

        String photo = "photo";
        String bucket = "bucket";

        AmazonRekognition rekognitionClient =
 AmazonRekognitionClientBuilder.defaultClient();

        DetectLabelsRequest request = new DetectLabelsRequest()
                .withImage(new Image().withS3Object(new
 S3Object().withName(photo).withBucket(bucket)))
                .withMaxLabels(10).withMinConfidence(75F);

        try {
```

```
            DetectLabelsResult result = rekognitionClient.detectLabels(request);
            List<Label> labels = result.getLabels();

            System.out.println("Detected labels for " + photo + "\n");
            for (Label label : labels) {
                System.out.println("Label: " + label.getName());
                System.out.println("Confidence: " +
 label.getConfidence().toString() + "\n");

                List<Instance> instances = label.getInstances();
                System.out.println("Instances of " + label.getName());
                if (instances.isEmpty()) {
                    System.out.println("  " + "None");
                } else {
                    for (Instance instance : instances) {
                        System.out.println("  Confidence: " +
 instance.getConfidence().toString());
                        System.out.println("  Bounding box: " +
 instance.getBoundingBox().toString());
                    }
                }
                System.out.println("Parent labels for " + label.getName() +
 ":");
                List<Parent> parents = label.getParents();
                if (parents.isEmpty()) {
                    System.out.println("  None");
                } else {
                    for (Parent parent : parents) {
                        System.out.println("  " + parent.getName());
                    }
                }
                System.out.println("--------------------");
                System.out.println();

            }
        } catch (AmazonRekognitionException e) {
            e.printStackTrace();
        }
    }
}
```

AWS CLI

Este ejemplo muestra la salida de JSON de la operación `detect-labels` de la CLI. Reemplace los valores de `bucket` y `photo` por los nombre del bucket de Amazon S3 y de la imagen que utilizó en el paso 2. Sustituya el valor de `profile-name` de por el nombre de su perfil de desarrollador.

```
aws rekognition detect-labels --image '{ "S3Object": { "Bucket": "bucket-name",
 "Name": "file-name" } }' \
--features GENERAL_LABELS IMAGE_PROPERTIES \
--settings '{"ImageProperties": {"MaxDominantColors":1}, {"GeneralLabels":
{"LabelInclusionFilters":["Cat"]}}}' \
--profile profile-name \
--region us-east-1
```

Si accede a la CLI en un dispositivo Windows, utilice comillas dobles en lugar de comillas simples y evite las comillas dobles internas con una barra invertida (es decir, \) para corregir cualquier error del analizador que pueda encontrar. Por ver un ejemplo, consulte lo siguiente:

```
aws rekognition detect-labels --image "{\"S3Object\":{\"Bucket\":\"bucket-name
\",\"Name\":\"file-name\"}}" --features GENERAL_LABELS IMAGE_PROPERTIES \
--settings "{\"GeneralLabels\":{\"LabelInclusionFilters\":[\"Car\"]}}" --profile
 profile-name --region us-east-1
```

Python

Este ejemplo muestra las etiquetas que se han detectado en la imagen de entrada. En la función `main`, cambie los valores de `bucket` y `photo` por los nombre del bucket de Amazon S3 y la imagen que utilizó en el paso 2. Sustituya el valor de `profile_name` en la línea que crea la sesión de Rekognition por el nombre de su perfil de desarrollador.

```
#Copyright 2018 Amazon.com, Inc. or its affiliates. All Rights Reserved.
#PDX-License-Identifier: MIT-0 (For details, see https://github.com/awsdocs/
amazon-rekognition-developer-guide/blob/master/LICENSE-SAMPLECODE.)

import boto3

def detect_labels(photo, bucket):
```

```
    session = boto3.Session(profile_name='profile-name')
    client = session.client('rekognition')

    response = client.detect_labels(Image={'S3Object':
{'Bucket':bucket,'Name':photo}},
    MaxLabels=10,
    # Uncomment to use image properties and filtration settings
    #Features=["GENERAL_LABELS", "IMAGE_PROPERTIES"],
    #Settings={"GeneralLabels": {"LabelInclusionFilters":["Cat"]},
    # "ImageProperties": {"MaxDominantColors":10}}
    )

    print('Detected labels for ' + photo)
    print()
    for label in response['Labels']:
        print("Label: " + label['Name'])
        print("Confidence: " + str(label['Confidence']))
        print("Instances:")

        for instance in label['Instances']:
            print(" Bounding box")
            print(" Top: " + str(instance['BoundingBox']['Top']))
            print(" Left: " + str(instance['BoundingBox']['Left']))
            print(" Width: " + str(instance['BoundingBox']['Width']))
            print(" Height: " + str(instance['BoundingBox']['Height']))
            print(" Confidence: " + str(instance['Confidence']))
            print()

        print("Parents:")
        for parent in label['Parents']:
           print(" " + parent['Name'])

        print("Aliases:")
        for alias in label['Aliases']:
            print(" " + alias['Name'])

            print("Categories:")
        for category in label['Categories']:
            print(" " + category['Name'])
            print("----------")
            print()

    if "ImageProperties" in str(response):
```

```
            print("Background:")
            print(response["ImageProperties"]["Background"])
            print()
            print("Foreground:")
            print(response["ImageProperties"]["Foreground"])
            print()
            print("Quality:")
            print(response["ImageProperties"]["Quality"])
            print()

        return len(response['Labels'])

def main():
    photo = 'photo-name'
    bucket = 'amzn-s3-demo-bucket'
    label_count = detect_labels(photo, bucket)
    print("Labels detected: " + str(label_count))

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

.NET

Este ejemplo muestra una lista de las etiquetas que se han detectado en la imagen de entrada. Reemplace los valores de `bucket` y `photo` por los nombre del bucket de Amazon S3 y de la imagen que utilizó en el paso 2.

```
//Copyright 2018 Amazon.com, Inc. or its affiliates. All Rights Reserved.
//PDX-License-Identifier: MIT-0 (For details, see https://github.com/awsdocs/
amazon-rekognition-developer-guide/blob/master/LICENSE-SAMPLECODE.)

using System;
using Amazon.Rekognition;
using Amazon.Rekognition.Model;

public class DetectLabels
{
    public static void Example()
    {
        String photo = "input.jpg";
        String bucket = "amzn-s3-demo-bucket";
```

```
        AmazonRekognitionClient rekognitionClient = new
 AmazonRekognitionClient();

        DetectLabelsRequest detectlabelsRequest = new DetectLabelsRequest()
        {
            Image = new Image()
            {
                S3Object = new S3Object()
                {
                    Name = photo,
                    Bucket = bucket
                },
            },
            MaxLabels = 10,
            MinConfidence = 75F
        };

        try
        {
            DetectLabelsResponse detectLabelsResponse =
 rekognitionClient.DetectLabels(detectlabelsRequest);
            Console.WriteLine("Detected labels for " + photo);
            foreach (Label label in detectLabelsResponse.Labels)
                Console.WriteLine("{0}: {1}", label.Name, label.Confidence);
        }
        catch (Exception e)
        {
            Console.WriteLine(e.Message);
        }
    }
}
```

Ruby

Este ejemplo muestra una lista de las etiquetas que se han detectado en la imagen de entrada. Reemplace los valores de `bucket` y `photo` por los nombre del bucket de Amazon S3 y de la imagen que utilizó en el paso 2.

```
  # Add to your Gemfile
  # gem 'aws-sdk-rekognition'
  require 'aws-sdk-rekognition'
```

```
 credentials = Aws::Credentials.new(
    ENV['AWS_ACCESS_KEY_ID'],
    ENV['AWS_SECRET_ACCESS_KEY']
 )
 bucket = 'bucket' # the bucket name without s3://
 photo  = 'photo' # the name of file
 client   = Aws::Rekognition::Client.new credentials: credentials
 attrs = {
   image: {
     s3_object: {
       bucket: bucket,
       name: photo
     },
   },
   max_labels: 10
 }
response = client.detect_labels attrs
puts "Detected labels for: #{photo}"
response.labels.each do |label|
  puts "Label:      #{label.name}"
  puts "Confidence: #{label.confidence}"
  puts "Instances:"
  label['instances'].each do |instance|
    box = instance['bounding_box']
    puts "  Bounding box:"
    puts "    Top:        #{box.top}"
    puts "    Left:       #{box.left}"
    puts "    Width:      #{box.width}"
    puts "    Height:     #{box.height}"
    puts "  Confidence: #{instance.confidence}"
  end
  puts "Parents:"
  label.parents.each do |parent|
    puts "  #{parent.name}"
  end
  puts "------------"
  puts ""
end
```

Node.js

Este ejemplo muestra una lista de las etiquetas que se han detectado en la imagen de entrada. Reemplace los valores de `bucket` y `photo` por los nombre del bucket de Amazon

S3 y de la imagen que utilizó en el paso 2. Sustituya el valor de `profile_name` en la línea que crea la sesión de Rekognition por el nombre de su perfil de desarrollador.

Si utiliza TypeScript definiciones, puede que necesite utilizarlas en `import AWS from 'aws-sdk'` lugar de`const AWS = require('aws-sdk')`, para ejecutar el programa con Node.js. Puede consultar el AWS SDK para Javascript si quiere obtener más información. En función de cómo tenga establecidas las opciones de configuración, es posible que también tenga que especificar su región con `AWS.config.update({region:`*`region`*`});`.

```
// Load the SDK
var AWS = require('aws-sdk');
const bucket = 'bucket-name' // the bucketname without s3://
const photo  = 'image-name' // the name of file

var credentials = new AWS.SharedIniFileCredentials({profile: 'profile-name'});
AWS.config.credentials = credentials;
AWS.config.update({region:'region-name'});

const client = new AWS.Rekognition();
const params = {
  Image: {
    S3Object: {
      Bucket: bucket,
      Name: photo
    },
  },
  MaxLabels: 10
}
client.detectLabels(params, function(err, response) {
  if (err) {
    console.log(err, err.stack); // if an error occurred
  } else {
    console.log(`Detected labels for: ${photo}`)
    response.Labels.forEach(label => {
      console.log(`Label:      ${label.Name}`)
      console.log(`Confidence: ${label.Confidence}`)
      console.log("Instances:")
      label.Instances.forEach(instance => {
        let box = instance.BoundingBox
        console.log("  Bounding box:")
```

```
        console.log(`    Top:        ${box.Top}`)
        console.log(`    Left:       ${box.Left}`)
        console.log(`    Width:      ${box.Width}`)
        console.log(`    Height:     ${box.Height}`)
        console.log(`  Confidence: ${instance.Confidence}`)
      })
      console.log("Parents:")
      label.Parents.forEach(parent => {
        console.log(`  ${parent.Name}`)
      })
      console.log("------------")
      console.log("")
    }) // for response.labels
  } // if
});
```

Java V2

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

```
//snippet-start:[rekognition.java2.detect_labels.import]
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.Image;
import software.amazon.awssdk.services.rekognition.model.DetectLabelsRequest;
import software.amazon.awssdk.services.rekognition.model.DetectLabelsResponse;
import software.amazon.awssdk.services.rekognition.model.Label;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import software.amazon.awssdk.services.rekognition.model.S3Object;
import java.util.List;

/**
* Before running this Java V2 code example, set up your development environment,
 including your credentials.
*
* For more information, see the following documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
```

```
*/
public class DetectLabels {

    public static void main(String[] args) {

        final String usage = "\n" +
            "Usage: " +
            "   <bucket> <image>\n\n" +
            "Where:\n" +
            "   bucket - The name of the Amazon S3 bucket that contains the
 image (for example, ,ImageBucket)." +
            "   image - The name of the image located in the Amazon S3 bucket
 (for example, Lake.png). \n\n";

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucket = args[0];
        String image = args[1];
        Region region = Region.US_WEST_2;
        RekognitionClient rekClient = RekognitionClient.builder()
            .region(region)
            .credentialsProvider(ProfileCredentialsProvider.create("profile-
name"))
            .build();

        getLabelsfromImage(rekClient, bucket, image);
        rekClient.close();
    }

    // snippet-start:[rekognition.java2.detect_labels_s3.main]
    public static void getLabelsfromImage(RekognitionClient rekClient, String
 bucket, String image) {

        try {
            S3Object s3Object = S3Object.builder()
                .bucket(bucket)
                .name(image)
                .build() ;

            Image myImage = Image.builder()
                .s3Object(s3Object)
```

```
                .build();

            DetectLabelsRequest detectLabelsRequest =
 DetectLabelsRequest.builder()
                .image(myImage)
                .maxLabels(10)
                .build();

            DetectLabelsResponse labelsResponse =
 rekClient.detectLabels(detectLabelsRequest);
            List<Label> labels = labelsResponse.labels();
            System.out.println("Detected labels for the given photo");
            for (Label label: labels) {
                System.out.println(label.name() + ": " +
 label.confidence().toString());
            }

        } catch (RekognitionException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
 // snippet-end:[rekognition.java2.detect_labels.main]
}
```

DetectLabels solicitud de operación

La entrada de `DetectLabel` es una imagen. En este ejemplo de entrada de JSON, la imagen de origen se carga desde un bucket de Amazon S3. `MaxLabels` es el número máximo de etiquetas que se devuelven en la respuesta. `MinConfidence` es el nivel mínimo de confianza que debe tener Amazon Rekognition Image en la precisión de la etiqueta detectada para que se devuelva en la respuesta.

Características le permite especificar una o más características de la imagen que desea que se devuelvan, lo que le permite seleccionar `GENERAL_LABELS` y `IMAGE_PROPERTIES`. Incluir `GENERAL_LABELS` devolverá las etiquetas detectadas en la imagen de entrada, mientras que incluir `IMAGE_PROPERTIES` le permitirá acceder al color y la calidad de la imagen.

La configuración le permite filtrar los artículos devueltos por sus características `GENERAL_LABELS` y `IMAGE_PROPERTIES`. Para las etiquetas, puede usar filtros inclusivos y exclusivos. También puede filtrar por etiquetas específicas, etiquetas individuales o por categoría de etiquetas:

- LabelInclusionFilters - Le permite especificar qué etiquetas desea incluir en la respuesta.
- LabelExclusionFilters - Le permite especificar qué etiquetas desea excluir de la respuesta.
- LabelCategoryInclusionFilters - Le permite especificar qué categorías de etiquetas desea incluir en la respuesta.
- LabelCategoryExclusionFilters - Le permite especificar qué categorías de etiquetas desea que se excluyan de la respuesta.

También puede combinar filtros inclusivos y exclusivos según sus necesidades, excluyendo algunas etiquetas o categorías e incluyendo otras.

`IMAGE_PROPERTIES` hace referencia a los colores dominantes y a los atributos de calidad de una imagen, como la nitidez, el brillo y el contraste. Al realizar la detección, `IMAGE_PROPERTIES` puede especificar el número máximo de colores dominantes que desea devolver (el valor predeterminado es 10) mediante el parámetro `MaxDominantColors`.

```
{
    "Image": {
        "S3Object": {
            "Bucket": "bucket",
            "Name": "input.jpg"
        }
    },
    "MaxLabels": 10,
    "MinConfidence": 75,
    "Features": [ "GENERAL_LABELS", "IMAGE_PROPERTIES" ],
    "Settings": {
        "GeneralLabels": {
            "LabelInclusionFilters": [<Label(s)>],
            "LabelExclusionFilters": [<Label(s)>],
            "LabelCategoryInclusionFilters": [<Category Name(s)>],
            "LabelCategoryExclusionFilters": [<Category Name(s)>]
        },
        "ImageProperties": {
            "MaxDominantColors":10
        }
    }
```

```
}
```

DetectLabels respuesta

La respuesta de `DetectLabels` es una matriz de las etiquetas detectadas en la imagen y el nivel de confianza por el que se detectan.

A continuación se muestra un ejemplo de respuesta de `DetectLabels`. El ejemplo de respuesta que aparece a continuación contiene una variedad de atributos devueltos para GENERAL_LABELS, entre los que se incluyen:

- Name: el nombre de la etiqueta detectada. En este ejemplo, la operación detectó un objeto con la etiqueta Teléfono móvil.
- Confidence: cada etiqueta tiene un nivel de confianza asociado. En este ejemplo, la confianza de la etiqueta era del 99,36 %.
- Parents: las etiquetas antecesoras de una etiqueta detectada. En este ejemplo, la etiqueta Teléfono móvil tiene una etiqueta principal llamada Teléfono.
- Aliases: información sobre los posibles alias de la etiqueta. En este ejemplo, la etiqueta Teléfono móvil tiene un posible alias de Móvil.
- Categories: la categoría de etiqueta a la que pertenece la etiqueta detectada. En este ejemplo, es Tecnología e Informática.

La respuesta para las etiquetas de objetos comunes incluye información del cuadro delimitador para localizar la etiqueta en la imagen de entrada. Por ejemplo, la etiqueta Person (persona) tiene una matriz de instancias que contiene dos cuadros delimitadores. Se trata de las ubicaciones de dos personas detectadas en la imagen.

La respuesta también incluye atributos relacionados con IMAGE_PROPERTIES. Los atributos que presenta la característica IMAGE_PROPERTIES son:

- Quality: información sobre la nitidez, el brillo y el contraste de la imagen de entrada, puntuada entre 0 y 100. La calidad se indica para toda la imagen y para el fondo y el primer plano de la imagen, si están disponibles. Sin embargo, el contraste solo se indica para toda la imagen, mientras que la nitidez y el brillo también se indican para el fondo y el primer plano.
- Dominant Color: conjunto de los colores dominantes de la imagen. Cada color dominante se describe con un nombre de color simplificado, una paleta de colores CSS, valores RGB y un código hexadecimal.

- Foreground: información sobre los colores, la nitidez y el brillo dominantes del primer plano de la imagen de entrada.
- Background: información sobre los colores, la nitidez y el brillo dominantes del fondo de la imagen de entrada.

Cuando GENERAL_LABELS e IMAGE_PROPERTIES se utilizan juntos como parámetros de entrada, Amazon Rekognition Image también devolverá los colores dominantes de los objetos con cuadros delimitadores.

El campo `LabelModelVersion` contiene el número de versión del modelo de detección que `DetectLabels` utiliza.


```
{

   "Labels": [
        {
            "Name": "Mobile Phone",
            "Parents": [
              {
                "Name": "Phone"
              }
            ],
            "Aliases": [
              {
                "Name": "Cell Phone"
              }
            ],
            "Categories": [
              {
                "Name": "Technology and Computing"
              }
            ],
            "Confidence": 99.9364013671875,
            "Instances": [
                {
                    "BoundingBox": {
                        "Width": 0.26779675483703613,
                        "Height": 0.8562285900115967,
                        "Left": 0.3604024350643158,
                        "Top": 0.09245597571134567,
                    }
```

```
                    "Confidence": 99.9364013671875,
                    "DominantColors": [
                    {
                "Red": 120,
                "Green": 137,
                "Blue": 132,
                "HexCode": "3A7432",
                "SimplifiedColor": "red",
                "CssColor": "fuscia",
                "PixelPercentage": 40.10
                    }
                        ],
                }
            ]
        }
    ],
    "ImageProperties": {
        "Quality": {
            "Brightness": 40,
            "Sharpness": 40,
            "Contrast": 24,
        },
        "DominantColors": [
            {
                "Red": 120,
                "Green": 137,
                "Blue": 132,
                "HexCode": "3A7432",
                "SimplifiedColor": "red",
                "CssColor": "fuscia",
                "PixelPercentage": 40.10
            }
        ],
        "Foreground": {
            "Quality": {
                "Brightness": 40,
                "Sharpness": 40,
            },
            "DominantColors": [
                {
                    "Red": 200,
                    "Green": 137,
                    "Blue": 132,
                    "HexCode": "3A7432",
```

```
                    "CSSColor": "",
                    "SimplifiedColor": "red",
                    "PixelPercentage": 30.70
                }
            ],
        }
        "Background": {
            "Quality": {
                "Brightness": 40,
                "Sharpness": 40,
            },
            "DominantColors": [
                {
                    "Red": 200,
                    "Green": 137,
                    "Blue": 132,
                    "HexCode": "3A7432",
                    "CSSColor": "",
                    "SimplifiedColor": "Red",
                    "PixelPercentage": 10.20
                }
            ],
        },
    },
    "LabelModelVersion": "3.0"
}
```

Transformar la DetectLabels respuesta

Cuando utilices la DetectLabels API, es posible que necesites que la estructura de respuesta de la API sea similar a la anterior, en la que tanto las etiquetas principales como los alias estaban contenidos en la misma lista.

El siguiente es un ejemplo de la respuesta actual de la API de: DetectLabels

```
"Labels": [
        {
            "Name": "Mobile Phone",
            "Confidence": 99.99717712402344,
            "Instances": [],
            "Parents": [
                {
                "Name": "Phone"
```

```
                }
             ],
            "Aliases": [
                {
                "Name": "Cell Phone"
                }
             ]
        }
 ]
```

En el siguiente ejemplo, se muestra la respuesta anterior de la DetectLabelsAPI:

```
"Labels": [
        {
            "Name": "Mobile Phone",
            "Confidence": 99.99717712402344,
            "Instances": [],
            "Parents": [
                {
                "Name": "Phone"
                }
             ]
         },
         {
            "Name": "Cell Phone",
            "Confidence": 99.99717712402344,
            "Instances": [],
            "Parents": [
                {
                "Name": "Phone"
                }
             ]
         },
]
```

Si es necesario, puede transformar la respuesta actual para que siga el formato de la respuesta anterior. Puede usar el siguiente código de ejemplo para transformar la última respuesta de la API en la estructura de respuesta de la API anterior:

Python

En el siguiente ejemplo de código, se muestra cómo transformar la respuesta actual de la DetectLabels API. En el ejemplo de código que aparece a continuación, puedes reemplazar el valor *EXAMPLE_INFERENCE_OUTPUT* de por el resultado de una DetectLabels operación que hayas ejecutado.

```
from copy import deepcopy

LABEL_KEY = "Labels"
ALIASES_KEY = "Aliases"
INSTANCE_KEY = "Instances"
NAME_KEY = "Name"

#Latest API response sample
EXAMPLE_INFERENCE_OUTPUT = {
    "Labels": [
        {
            "Name": "Mobile Phone",
            "Confidence": 97.530106,
            "Categories": [
                {
                    "Name": "Technology and Computing"
                }
            ],
            "Aliases": [
                {
                    "Name": "Cell Phone"
                }
            ],
            "Instances":[
                {
                    "BoundingBox":{
                        "Height":0.1549897,
                        "Width":0.07747964,
                        "Top":0.50858885,
                        "Left":0.00018205095
                    },
                    "Confidence":98.401276
                }
            ]
        },
        {
```

```
            "Name": "Urban",
            "Confidence": 99.99982,
            "Categories": [
                "Colors and Visual Composition"
            ]
        }
    ]
}

def expand_aliases(inferenceOutputsWithAliases):

    if LABEL_KEY in inferenceOutputsWithAliases:
        expandInferenceOutputs = []
        for primaryLabelDict in inferenceOutputsWithAliases[LABEL_KEY]:
            if ALIASES_KEY in primaryLabelDict:
                for alias in primaryLabelDict[ALIASES_KEY]:
                    aliasLabelDict = deepcopy(primaryLabelDict)
                    aliasLabelDict[NAME_KEY] = alias[NAME_KEY]
                    del aliasLabelDict[ALIASES_KEY]
                    if INSTANCE_KEY in aliasLabelDict:
                        del aliasLabelDict[INSTANCE_KEY]
                    expandInferenceOutputs.append(aliasLabelDict)

        inferenceOutputsWithAliases[LABEL_KEY].extend(expandInferenceOutputs)

    return inferenceOutputsWithAliases


if __name__ == "__main__":

    outputWithExpandAliases = expand_aliases(EXAMPLE_INFERENCE_OUTPUT)
    print(outputWithExpandAliases)
```

A continuación se ofrece un ejemplo de respuesta transformada:

```
#Output example after the transformation
{
    "Labels": [
        {
            "Name": "Mobile Phone",
            "Confidence": 97.530106,
            "Categories": [
```

```
                {
                    "Name": "Technology and Computing"
                }
            ],
            "Aliases": [
                {
                    "Name": "Cell Phone"
                }
            ],
            "Instances":[
                {
                    "BoundingBox":{
                        "Height":0.1549897,
                        "Width":0.07747964,
                        "Top":0.50858885,
                        "Left":0.00018205095
                    },
                    "Confidence":98.401276
                }
            ]
        },
        {
            "Name": "Cell Phone",
            "Confidence": 97.530106,
            "Categories": [
                {
                    "Name": "Technology and Computing"
                }
            ],
            "Instances":[]
        },
        {
            "Name": "Urban",
            "Confidence": 99.99982,
            "Categories": [
                "Colors and Visual Composition"
            ]
        }
    ]
}
```

Detección de etiquetas en un vídeo

Amazon Rekognition Video puede detectar etiquetas (objetos y conceptos) y la hora en que se detecta una etiqueta en un vídeo. Para ver un ejemplo de código del SDK, consulte Análisis de un vídeo almacenado en un bucket de Amazon S3 con Java o Python (SDK). Para ver un AWS CLI ejemplo, consulteAnalizar un vídeo con el AWS Command Line Interface.

La detección de etiquetas de Amazon Rekognition Video es una operación asíncrona. Para iniciar la detección de etiquetas en un vídeo, llame StartLabelDetection.

Amazon Rekognition Video publica el estado de finalización de una operación de análisis de vídeo en un tema de Amazon Simple Notification Service. Si el análisis de vídeo es correcto, llame a GetLabelDetection para obtener las etiquetas detectadas. Para obtener información sobre cómo llamar a operaciones de la API de análisis de vídeo, consulte Cómo llamar a las operaciones de Amazon Rekognition Video.

StartLabelDetection Solicita

A continuación, se muestra un ejemplo para la operación `StartLabelDetection`. Usted proporciona a la operación `StartLabelDetection` un vídeo almacenado en un bucket de Amazon S3. En la solicitud JSON de ejemplo, se especifican el bucket de Amazon S3 y el nombre del vídeo, junto con`MinConfidence`, `Features`, `Settings` y `NotificationChannel`.

`MinConfidence` es el nivel mínimo de confianza que debe tener Amazon Rekognition Video en la precisión de la etiqueta detectada, o una instancia de cuadro delimitador (si se detecta), para que se devuelva en la respuesta.

Con `Features`, puede especificar que desea que se devuelva GENERAL_LABELS como parte de la respuesta.

Con `Settings`, puede filtrar los artículos devueltos para GENERAL_LABELS. Para las etiquetas, puede usar filtros inclusivos y exclusivos. También puede filtrar por etiquetas específicas, etiquetas individuales o por categoría de etiquetas:

- `LabelInclusionFilters`: se utiliza para especificar qué etiquetas desea incluir en la respuesta
- `LabelExclusionFilters`: se utiliza para especificar qué etiquetas desea excluir de la respuesta.
- `LabelCategoryInclusionFilters`: se utiliza para especificar qué categorías de etiquetas desea incluir en la respuesta.

- `LabelCategoryExclusionFilters`: se utiliza para especificar qué categorías de etiquetas desea excluir de la respuesta.

También puede combinar filtros inclusivos y exclusivos según sus necesidades, excluyendo algunas etiquetas o categorías e incluyendo otras.

`NotificationChannel` es el ARN del tema de Amazon SNS en el que desea que Amazon Rekognition Video publique el estado de finalización de la operación de detección de etiquetas. Si utiliza la política de permisos `AmazonRekognitionServiceRole`, el tema de Amazon SNS debe tener un nombre que comience por Rekognition.

El siguiente es un ejemplo de solicitud `StartLabelDetection` en formato JSON, que incluye filtros:

```
{
    "ClientRequestToken": "5a6e690e-c750-460a-9d59-c992e0ec8638",
    "JobTag": "5a6e690e-c750-460a-9d59-c992e0ec8638",
    "Video": {
        "S3Object": {
            "Bucket": "bucket",
            "Name": "video.mp4"
        }
     },
     "Features": ["GENERAL_LABELS"],
     "MinConfidence": 75,
     "Settings": {
         "GeneralLabels": {
             "LabelInclusionFilters": ["Cat", "Dog"],
             "LabelExclusionFilters": ["Tiger"],
             "LabelCategoryInclusionFilters": ["Animals and Pets"],
             "LabelCategoryExclusionFilters": ["Popular Landmark"]
         }
     },
     "NotificationChannel": {
         "RoleArn": "arn:aws:iam::012345678910:role/SNSAccessRole",
         "SNSTopicArn": "arn:aws:sns:us-east-1:012345678910:notification-topic",
     }
}
```

GetLabelDetection Respuesta a la operación

`GetLabelDetection` devuelve una matriz (`Labels`) que contiene información sobre las etiquetas detectadas en el vídeo. La matriz se puede ordenar por tiempo o por la etiqueta detectada al especificar el parámetro `SortBy`. También puede seleccionar cómo se agregan los elementos de respuesta mediante el parámetro `AggregateBy`.

El siguiente ejemplo es la respuesta JSON de `GetLabelDetection`. En la respuesta, tenga en cuenta lo siguiente:

- Orden de clasificación: la matriz de etiquetas devueltas está ordenada por tiempo. Para ordenar por etiqueta, especifique `NAME` en el parámetro de entrada `SortBy` para `GetLabelDetection`. Si la etiqueta aparece varias veces en el vídeo, habrá varias instancias del elemento (LabelDetection). El orden de clasificación predeterminado es `TIMESTAMP`, mientras que el orden de clasificación secundario es `NAME`.
- Información de etiqueta: el elemento de matriz `LabelDetection` contiene un objeto (Label), que a su vez contiene el nombre de la etiqueta y la confianza que Amazon Rekognition tiene en la precisión de la etiqueta detectada. Los objetos `Label` también contienen una taxonomía jerárquica de etiquetas e información de los cuadros delimitadores de las etiquetas comunes. `Timestamp` es el tiempo de detección de la etiqueta, definido como el número de milisegundos transcurrido desde el comienzo del vídeo.

  También se devuelve información sobre las categorías o alias asociados a una etiqueta. Para los resultados agregados por vídeo `SEGMENTS`, se devuelven las estructuras `StartTimestampMillis`, `EndTimestampMillis` y `DurationMillis`, que definen la hora de inicio, la hora de finalización y la duración de un segmento, respectivamente.
- Agregación: especifica cómo se agregan los resultados cuando se devuelven. El valor predeterminado es agregar por `TIMESTAMPS`. También puede optar por agregar por `SEGMENTS`, lo que agrega los resultados en un intervalo de tiempo. Si se agrega por `SEGMENTS`, no se devuelve la información sobre las instancias detectadas con recuadros delimitadores. Solo se devuelven las etiquetas detectadas durante los segmentos.
- Información de paginación: el ejemplo muestra una página de información de detección de etiqueta. Puede especificar la cantidad de objetos `LabelDetection` que se van a devolver en el parámetro de entrada `MaxResults` para `GetLabelDetection`. Si existen más resultados que `MaxResults`, `GetLabelDetection` devuelve un token (`NextToken`) que se utiliza para obtener la siguiente página de resultados. Para obtener más información, consulte Obtención de los resultados del análisis de Amazon Rekognition Video.

- Información de vídeo: la respuesta incluye información acerca del formato de vídeo (`VideoMetadata`) en cada página de información devuelta por `GetLabelDetection`.

El siguiente es un ejemplo de GetLabelDetection respuesta en formato JSON con agregación mediante TIMESTAMPS:

```
{
    "JobStatus": "SUCCEEDED",
    "LabelModelVersion": "3.0",
    "Labels": [
        {
            "Timestamp": 1000,
            "Label": {
                "Name": "Car",
                "Categories": [
                  {
                    "Name": "Vehicles and Automotive"
                  }
                ],
                "Aliases": [
                  {
                    "Name": "Automobile"
                  }
                ],
                "Parents": [
                  {
                    "Name": "Vehicle"
                  }
                ],
                "Confidence": 99.9364013671875, // Classification confidence
                "Instances": [
                    {
                        "BoundingBox": {
                            "Width": 0.26779675483703613,
                            "Height": 0.8562285900115967,
                            "Left": 0.3604024350643158,
                            "Top": 0.09245597571134567
                        },
                        "Confidence": 99.9364013671875 // Detection confidence
                    }
                ]
            }
```

```
        },
        {
            "Timestamp": 1000,
            "Label": {
                "Name": "Cup",
                "Categories": [
                  {
                    "Name": "Kitchen and Dining"
                  }
                ],
                "Aliases": [
                  {
                    "Name": "Mug"
                  }
                ],
                "Parents": [],
                "Confidence": 99.9364013671875, // Classification confidence
                "Instances": [
                    {
                        "BoundingBox": {
                            "Width": 0.26779675483703613,
                            "Height": 0.8562285900115967,
                            "Left": 0.3604024350643158,
                            "Top": 0.09245597571134567
                        },
                        "Confidence": 99.9364013671875 // Detection confidence
                    }
                ]
            }
        },
        {
            "Timestamp": 2000,
            "Label": {
                "Name": "Kangaroo",
                "Categories": [
                  {
                    "Name": "Animals and Pets"
                  }
                ],
                "Aliases": [
                  {
                    "Name": "Wallaby"
                  }
                ],
```

```
                "Parents": [
                  {
                    "Name": "Mammal"
                  }
                ],
                "Confidence": 99.9364013671875,
                "Instances": [
                    {
                        "BoundingBox": {
                            "Width": 0.26779675483703613,
                            "Height": 0.8562285900115967,
                            "Left": 0.3604024350643158,
                            "Top": 0.09245597571134567,
                        },
                        "Confidence": 99.9364013671875
                    }
                ]
            }
        },
        {
            "Timestamp": 4000,
            "Label": {
                "Name": "Bicycle",
                "Categories": [
                  {
                    "Name": "Hobbies and Interests"
                  }
                ],
                "Aliases": [
                  {
                    "Name": "Bike"
                  }
                ],
                "Parents": [
                  {
                    "Name": "Vehicle"
                  }
                ],
                "Confidence": 99.9364013671875,
                "Instances": [
                    {
                        "BoundingBox": {
                            "Width": 0.26779675483703613,
                            "Height": 0.8562285900115967,
```

```
                        "Left": 0.3604024350643158,
                        "Top": 0.09245597571134567
                    },
                    "Confidence": 99.9364013671875
                }
            ]
        }
    }
],
"VideoMetadata": {
    "ColorRange": "FULL",
    "DurationMillis": 5000,
    "Format": "MP4",
    "FrameWidth": 1280,
    "FrameHeight": 720,
    "FrameRate": 24
}
}
```

El siguiente es un ejemplo de GetLabelDetection respuesta en formato JSON con agregación por SEGMENTOS:

```
{
    "JobStatus": "SUCCEEDED",
    "LabelModelVersion": "3.0",
    "Labels": [
        {
            "StartTimestampMillis": 225,
            "EndTimestampMillis": 3578,
            "DurationMillis": 3353,
            "Label": {
                "Name": "Car",
                "Categories": [
                  {
                    "Name": "Vehicles and Automotive"
                  }
                ],
                "Aliases": [
                  {
                    "Name": "Automobile"
                  }
                ],
                "Parents": [
```

```
                {
                  "Name": "Vehicle"
                }
              ],
              "Confidence": 99.9364013671875 // Maximum confidence score for Segment
mode
            }
        },
        {
            "StartTimestampMillis": 7578,
            "EndTimestampMillis": 12371,
            "DurationMillis": 4793,
            "Label": {
                "Name": "Kangaroo",
                "Categories": [
                  {
                    "Name": "Animals and Pets"
                  }
                ],
                "Aliases": [
                  {
                    "Name": "Wallaby"
                  }
                ],
                "Parents": [
                  {
                    "Name": "Mammal"
                  }
                ],
                "Confidence": 99.9364013671875
            }
        },
        {
            "StartTimestampMillis": 22225,
            "EndTimestampMillis": 22578,
            "DurationMillis": 2353,
            "Label": {
                "Name": "Bicycle",
                "Categories": [
                  {
                    "Name": "Hobbies and Interests"
                  }
                ],
                "Aliases": [
```

```
                {
                  "Name": "Bike"
                }
              ],
              "Parents": [
                {
                  "Name": "Vehicle"
                }
              ],
              "Confidence": 99.9364013671875
            }
        }
    ],
    "VideoMetadata": {
        "ColorRange": "FULL",
        "DurationMillis": 5000,
        "Format": "MP4",
        "FrameWidth": 1280,
        "FrameHeight": 720,
        "FrameRate": 24
    }
}
```

Transformando la GetLabelDetection respuesta

Al recuperar los resultados de la operación de la GetLabelDetection API, es posible que necesite que la estructura de respuesta imite la estructura de respuesta de la API anterior, en la que tanto las etiquetas principales como los alias estaban contenidos en la misma lista.

El ejemplo de respuesta de JSON que se encuentra en la sección anterior muestra la forma actual de la respuesta de la API de. GetLabelDetection

En el siguiente ejemplo, se muestra la respuesta anterior de la GetLabelDetection API:

```
{
    "Labels": [
        {
            "Timestamp": 0,
            "Label": {
                "Instances": [],
                "Confidence": 60.51791763305664,
                "Parents": [],
                "Name": "Leaf"
```

```
            }
        },
        {
            "Timestamp": 0,
            "Label": {
                "Instances": [],
                "Confidence": 99.53411102294922,
                "Parents": [],
                "Name": "Human"
            }
        },
        {
            "Timestamp": 0,
            "Label": {
                "Instances": [
                    {
                        "BoundingBox": {
                            "Width": 0.11109819263219833,
                            "Top": 0.08098889887332916,
                            "Left": 0.8881205320358276,
                            "Height": 0.9073750972747803
                        },
                        "Confidence": 99.5831298828125
                    },
                    {
                        "BoundingBox": {
                            "Width": 0.1268676072359085,
                            "Top": 0.14018426835536957,
                            "Left": 0.0003282368124928324,
                            "Height": 0.7993982434272766
                        },
                        "Confidence": 99.46029663085938
                    }
                ],
                "Confidence": 99.63411102294922,
                "Parents": [],
                "Name": "Person"
            }
        },
        .
        .
        .

        {
```

```
            "Timestamp": 166,
            "Label": {
                "Instances": [],
                "Confidence": 73.6471176147461,
                "Parents": [
                    {
                        "Name": "Clothing"
                    }
                ],
                "Name": "Sleeve"
            }
        }

    ],
    "LabelModelVersion": "2.0",
    "JobStatus": "SUCCEEDED",
    "VideoMetadata": {
        "Format": "QuickTime / MOV",
        "FrameRate": 23.976024627685547,
        "Codec": "h264",
        "DurationMillis": 5005,
        "FrameHeight": 674,
        "FrameWidth": 1280
    }
}
```

Si es necesario, puede transformar la respuesta actual para que siga el formato de la respuesta anterior. Puede usar el siguiente código de ejemplo para transformar la última respuesta de la API en la estructura de respuesta de la API anterior:

```
from copy import deepcopy

VIDEO_LABEL_KEY = "Labels"
LABEL_KEY = "Label"
ALIASES_KEY = "Aliases"
INSTANCE_KEY = "Instances"
NAME_KEY = "Name"

#Latest API response sample for AggregatedBy SEGMENTS
EXAMPLE_SEGMENT_OUTPUT = {
    "Labels": [
        {
            "Timestamp": 0,
```

```
            "Label":{
                "Name": "Person",
                "Confidence": 97.530106,
                "Parents": [],
                "Aliases": [
                    {
                        "Name": "Human"
                    },
                ],
                "Categories": [
                    {
                        "Name": "Person Description"
                    }
                ],
            },
            "StartTimestampMillis": 0,
            "EndTimestampMillis": 500666,
            "DurationMillis": 500666
        },
        {
            "Timestamp": 6400,
            "Label": {
                "Name": "Leaf",
                "Confidence": 89.77790069580078,
                "Parents": [
                    {
                        "Name": "Plant"
                    }
                ],
                "Aliases": [],
                "Categories": [
                    {
                        "Name": "Plants and Flowers"
                    }
                ],

            },
            "StartTimestampMillis": 6400,
            "EndTimestampMillis": 8200,
            "DurationMillis": 1800
        },
    ]
}
```

```
#Output example after the transformation for AggregatedBy SEGMENTS
EXPECTED_EXPANDED_SEGMENT_OUTPUT = {
    "Labels": [
        {
            "Timestamp": 0,
            "Label":{
                "Name": "Person",
                "Confidence": 97.530106,
                "Parents": [],
                "Aliases": [
                    {
                        "Name": "Human"
                    },
                ],
                "Categories": [
                    {
                        "Name": "Person Description"
                    }
                ],
            },
            "StartTimestampMillis": 0,
            "EndTimestampMillis": 500666,
            "DurationMillis": 500666
        },
        {
            "Timestamp": 6400,
            "Label": {
                "Name": "Leaf",
                "Confidence": 89.77790069580078,
                "Parents": [
                    {
                        "Name": "Plant"
                    }
                ],
                "Aliases": [],
                "Categories": [
                    {
                        "Name": "Plants and Flowers"
                    }
                ],

            },
            "StartTimestampMillis": 6400,
            "EndTimestampMillis": 8200,
```

```
            "DurationMillis": 1800
        },
        {
            "Timestamp": 0,
            "Label":{
                "Name": "Human",
                "Confidence": 97.530106,
                "Parents": [],
                "Categories": [
                    {
                        "Name": "Person Description"
                    }
                ],
            },
            "StartTimestampMillis": 0,
            "EndTimestampMillis": 500666,
            "DurationMillis": 500666
        },
    ]
}

#Latest API response sample for AggregatedBy TIMESTAMPS
EXAMPLE_TIMESTAMP_OUTPUT = {
    "Labels": [
        {
            "Timestamp": 0,
            "Label": {
                "Name": "Person",
                "Confidence": 97.530106,
                "Instances": [
                    {
                        "BoundingBox": {
                            "Height": 0.1549897,
                            "Width": 0.07747964,
                            "Top": 0.50858885,
                            "Left": 0.00018205095
                        },
                        "Confidence": 97.530106
                    },
                ],
                "Parents": [],
                "Aliases": [
                    {
                        "Name": "Human"
```

```
                    },
                ],
                "Categories": [
                    {
                        "Name": "Person Description"
                    }
                ],
            },
        },
        {
            "Timestamp": 6400,
            "Label": {
                "Name": "Leaf",
                "Confidence": 89.77790069580078,
                "Instances": [],
                "Parents": [
                    {
                        "Name": "Plant"
                    }
                ],
                "Aliases": [],
                "Categories": [
                    {
                        "Name": "Plants and Flowers"
                    }
                ],
            },
        },
    ]
}

#Output example after the transformation for AggregatedBy TIMESTAMPS
EXPECTED_EXPANDED_TIMESTAMP_OUTPUT = {
    "Labels": [
        {
            "Timestamp": 0,
            "Label": {
                "Name": "Person",
                "Confidence": 97.530106,
                "Instances": [
                    {
                        "BoundingBox": {
                            "Height": 0.1549897,
                            "Width": 0.07747964,
```

```
                        "Top": 0.50858885,
                        "Left": 0.00018205095
                    },
                    "Confidence": 97.530106
                },
            ],
            "Parents": [],
            "Aliases": [
                {
                    "Name": "Human"
                },
            ],
            "Categories": [
                {
                    "Name": "Person Description"
                }
            ],
        },
    },
    {
        "Timestamp": 6400,
        "Label": {
            "Name": "Leaf",
            "Confidence": 89.77790069580078,
            "Instances": [],
            "Parents": [
                {
                    "Name": "Plant"
                }
            ],
            "Aliases": [],
            "Categories": [
                {
                    "Name": "Plants and Flowers"
                }
            ],
        },
    },
    {
        "Timestamp": 0,
        "Label": {
            "Name": "Human",
            "Confidence": 97.530106,
            "Parents": [],
```

```
                "Categories": [
                    {
                        "Name": "Person Description"
                    }
                ],
            },
        },
    ]
}

def expand_aliases(inferenceOutputsWithAliases):

    if VIDEO_LABEL_KEY in inferenceOutputsWithAliases:
        expandInferenceOutputs = []
        for segmentLabelDict in inferenceOutputsWithAliases[VIDEO_LABEL_KEY]:
            primaryLabelDict = segmentLabelDict[LABEL_KEY]
            if ALIASES_KEY in primaryLabelDict:
                for alias in primaryLabelDict[ALIASES_KEY]:
                    aliasLabelDict = deepcopy(segmentLabelDict)
                    aliasLabelDict[LABEL_KEY][NAME_KEY] = alias[NAME_KEY]
                    del aliasLabelDict[LABEL_KEY][ALIASES_KEY]
                    if INSTANCE_KEY in aliasLabelDict[LABEL_KEY]:
                        del aliasLabelDict[LABEL_KEY][INSTANCE_KEY]
                    expandInferenceOutputs.append(aliasLabelDict)

        inferenceOutputsWithAliases[VIDEO_LABEL_KEY].extend(expandInferenceOutputs)

    return inferenceOutputsWithAliases


if __name__ == "__main__":

    segmentOutputWithExpandAliases = expand_aliases(EXAMPLE_SEGMENT_OUTPUT)
    assert segmentOutputWithExpandAliases == EXPECTED_EXPANDED_SEGMENT_OUTPUT

    timestampOutputWithExpandAliases = expand_aliases(EXAMPLE_TIMESTAMP_OUTPUT)
    assert timestampOutputWithExpandAliases == EXPECTED_EXPANDED_TIMESTAMP_OUTPUT
```

Detección de etiquetas en eventos de vídeo en streaming

Puede utilizar Amazon Rekognition Video para detectar etiquetas en la transmisión de vídeo. Para ello, debe crear un procesador de transmisión (CreateStreamProcessor) para iniciar y gestionar el análisis de la transmisión de vídeo.

Amazon Rekognition Video utiliza Amazon Kinesis Video Streams para recibir y procesar una transmisión de vídeo. Cuando crea el procesador de secuencias, elige lo que quiere que detecte. Puede elegir personas, paquetes y mascotas o personas y paquetes. Los resultados del análisis se envían en su bucket de Amazon S3 y en notificaciones de Amazon SNS. Tenga en cuenta que Amazon Rekognition Video detecta la presencia de una persona en el vídeo, pero no detecta si se trata de una persona específica. Para buscar un rostro de una colección en un vídeo en streaming, consulte the section called “Búsqueda de rostros en una colección en streaming de vídeo”.

Para utilizar Amazon Rekognition Video con la transmisión de vídeo, la aplicación requiere lo siguiente:

- Una transmisión de vídeo de Kinesis para enviar vídeo en streaming a Amazon Rekognition Video. Para obtener más información, consulte la Guía para desarrolladores de Amazon Kinesis Vídeo Streams.
- Un procesador de streaming de Amazon Rekognition Video para administrar el análisis del vídeo en streaming. Para obtener más información, consulte Descripción general de las operaciones del procesador de transmisión de Amazon Rekognition Video.
- Un bucket de Amazon S3. Amazon Rekognition Video publica los resultados de sesión en el bucket de S3. El resultado incluye el fotograma de la imagen en el que se detectó por primera vez a una persona u objeto de interés. Debe ser el propietario del bucket de S3.
- Un tema de Amazon SNS sobre el que Amazon Rekognition Video publica alertas inteligentes y un resumen. end-of-session

Temas

- Configuración de los recursos de Amazon Rekognition Video y Amazon Kinesis
- Llamada a operaciones de detección de etiquetas para la transmisión de eventos de vídeo

Configuración de los recursos de Amazon Rekognition Video y Amazon Kinesis

Los siguientes procedimientos describen los pasos que debe seguir para aprovisionar la transmisión de vídeo de Kinesis y otros recursos que se utilizan para detectar etiquetas en una transmisión de vídeo.

Requisitos previos

Para ejecutar este procedimiento, AWS SDK para Java debe estar instalado. Para obtener más información, consulte Introducción a Amazon Rekognition. La cuenta de AWS que utilice requiere permisos de acceso a la API Amazon Rekognition. Para obtener más información, consulte Acciones definidas por Amazon Rekognition en la Guía del usuario de IAM.

Para detectar etiquetas en una transmisión de vídeo (AWS SDK)

1. Crear un bucket de Amazon S3. Anote el nombre del bucket y los prefijos clave que desea utilizar. Usará esta información más tarde.
2. Crear un tema de Amazon SNS Puede usarlo para recibir notificaciones cuando se detecta por primera vez un objeto de interés en la transmisión de vídeo. Anote el nombre de recurso de Amazon (ARN) para el tema. Para obtener más información, consulte Creación de un tema de Amazon SNS en la Guía para desarrolladores de Amazon SNS.
3. Suscriba un punto de conexión a un tema de Amazon SNS. Para obtener más información, consulte Suscripción a un tema de Amazon SNS en la Guía del desarrollador de Amazon SNS.
4. Cree una transmisión de vídeo de Kinesis y anote el Nombre de recurso de Amazon (ARN) de la transmisión.
5. Si no lo ha hecho aún, cree un rol de servicio de IAM para otorgar a Amazon Rekognition Video acceso a sus transmisiones de vídeo de Kinesis, su bucket de S3 y sus temas de Amazon SNS. Para obtener más información, consulte Permiten el acceso a los procesadores de flujo con detección de etiquetas.

A continuación, puede crear el procesador de transmisión por detección de etiquetas e iniciar el procesador de transmisión que haya elegido.

> **Note**
>
> Inicie el procesador de transmisión solo después de comprobar que puede introducir contenido multimedia en la transmisión de vídeo de Kinesis.

Orientación y configuración de la cámara

Amazon Rekognition Video Streaming Video Events son compatibles con todas las cámaras compatibles con Kinesis Video Streams. Para obtener los mejores resultados, recomendamos colocar la cámara entre 0 y 45 grados del suelo. La cámara debe estar en posición vertical canónica. Por ejemplo, si hay una persona en el fotograma, la persona debe estar orientada verticalmente y la cabeza de la persona debe estar más alta en el fotograma que los pies.

Permiten el acceso a los procesadores de flujo con detección de etiquetas

Utiliza un rol de servicio AWS Identity and Access Management (IAM) para dar a Amazon Rekognition Video acceso de lectura a las transmisiones de vídeo de Kinesis. Para ello, utilice los roles de IAM para dar a Amazon Rekognition Video acceso a su bucket de Amazon S3 y a un tema de Amazon SNS.

Puede crear una política de permisos que permita a Amazon Rekognition Video acceder a un tema existente de Amazon SNS, a un bucket de Amazon S3 y a la transmisión de vídeo de Kinesis. Para ver un step-by-step procedimiento en el que se utiliza, consulte. AWS CLIthe section called “AWS CLI comandos para configurar una función de IAM de detección de etiquetas”

Para dar acceso a Amazon Rekognition Video a los recursos para la detección de etiquetas

1. Cree una nueva política de permisos con el editor de políticas de JSON de IAM y utilice la política siguiente. Sustituya `kvs-stream-name` por el nombre de la transmisión de vídeo de Kinesis, `topicarn` por el nombre de recurso de Amazon (ARN) del tema de Amazon SNS que desee utilizar y `bucket-name` por el nombre del bucket de Amazon S3.
2. Cree un rol de servicio de IAM o actualice un rol de servicio de IAM existente. Utilice la siguiente información para crear el rol de servicio de IAM:
    1. Elija Rekognition para el nombre del servicio.
    2. Elija Rekognition para el caso de uso del rol de servicio.
    3. Adjunte la política de permisos que ha creado en el paso 1.

3. Anote el ARN del rol de servicio. Lo necesita para crear el procesador de transmisión antes de realizar las operaciones de análisis de vídeo.
4. (Opcional) Si utiliza su propia AWS KMS clave para cifrar los datos enviados a su bucket de S3, debe añadir la siguiente declaración a la función de IAM. (Este es el rol de IAM que creó para la política de claves, que corresponde a la clave administrada por el cliente que desea usar).

```
            {
                        "Sid": "Allow use of the key by label detection Role",
                        "Effect": "Allow",
                        "Principal": {
                            "AWS":
"arn:aws:iam:::role/REPLACE_WITH_LABEL_DETECTION_ROLE_CREATED"
                        },
                        "Action": [
                            "kms:Decrypt",
                            "kms:GenerateDataKey*"
                        ],
                        "Resource": "*"
            }
```

AWS CLI comandos para configurar una función de IAM de detección de etiquetas

Si aún no lo ha hecho, configúrelo y configúrelo AWS CLI con sus credenciales.

Introduzca los siguientes comandos en el AWS CLI para configurar un rol de IAM con los permisos necesarios para la detección de etiquetas.

1. `export IAM_ROLE_NAME=labels-test-role`
2. `export AWS_REGION=us-east-1`
3. Cree un archivo de política de relaciones de confianza (por ejemplo, assume-role-rekognition .json) con el siguiente contenido.

   JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
```

```
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "Service": "rekognition.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

4. `aws iam create-role --role-name $IAM_ROLE_NAME --assume-role-policy-document file://path-to-assume-role-rekognition.json --region $AWS_REGION`
5. `aws iam attach-role-policy --role-name $IAM_ROLE_NAME --policy-arn "arn:aws:iam::aws:policy/service-role/AmazonRekognitionServiceRole" --region $AWS_REGION`
6. Si el nombre del tema de SNS con el que deseas recibir notificaciones no comienza con el prefijo «AmazonRekognition», agrega la siguiente política:

   `aws iam attach-role-policy --role-name $IAM_ROLE_NAME --policy-arn "arn:aws:iam::aws:policy/AmazonSNSFullAccess" --region $AWS_REGION`
7. Si utiliza su propia clave de AWS KMS para cifrar los datos que se envían a su bucket de Amazon S3, actualice la política de claves de la clave administrada por el cliente que desea utilizar.

   a. Cree un archivo kms_key_policy.json que tenga el siguiente contenido:

```
{
"Sid": "Allow use of the key by label detection Role",
"Effect": "Allow",
"Principal": {
"AWS": "arn:aws:iam:::role/REPLACE_WITH_IAM_ROLE_NAME_CREATED"
},
"Action": [
"kms:Encrypt",
"kms:GenerateDataKey*"
],
"Resource": "*"
}
```

b. `export KMS_KEY_ID=labels-kms-key-id`. Sustituya KMS_KEY_ID por el identificador de clave de KMS que creó.

c. `aws kms put-key-policy --policy-name default --key-id $KMS_KEY_ID --policy file://path-to-kms-key-policy.json`

Llamada a operaciones de detección de etiquetas para la transmisión de eventos de vídeo

Amazon Rekognition Video puede detectar personas u objetos relevantes en una transmisión de vídeo y notificarle cuando los detecte. Cuando cree un procesador de transmisión de detección de etiquetas, elija qué etiquetas quiere que Amazon Rekognition Video detecte. Pueden ser personas, paquetes y mascotas, o personas, paquetes, mascotas. Elija solo las etiquetas específicas que desee detectar. De esta forma, las únicas etiquetas relevantes crean notificaciones. Puede configurar las opciones para determinar cuándo almacenar la información de vídeo y, a continuación, realizar un procesamiento adicional en función de las etiquetas que se detecten en el fotograma.

Tras configurar los recursos, el proceso para detectar etiquetas en una transmisión de vídeo es el siguiente:

1. Creación del procesador de streaming
2. Iniciar el procesador de streaming
3. Si se detecta un objeto de interés, recibirá una notificación de Amazon SNS la primera vez que aparezca cada objeto de interés.
4. El procesador de transmisiones se detiene cuando finaliza el tiempo especificado en `MaxDurationInSeconds`.
5. Recibirá una notificación final de Amazon SNS con un resumen del evento.
6. Amazon Rekognition Video publica un resumen detallado de la sesión en su bucket de S3.

Temas

- Creación del procesador de flujo de detección de etiquetas de Amazon Rekognition Video
- Iniciar el procesador de flujo de detección de etiquetas de Amazon Rekognition Video
- Analizar los resultados de la detección de etiquetas

Creación del procesador de flujo de detección de etiquetas de Amazon Rekognition Video

Antes de poder analizar un vídeo en streaming, debe crear un procesador de streaming Amazon Rekognition Video (). CreateStreamProcessor

Si desea crear un procesador de transmisiones para detectar etiquetas de interés y personas, proporcione como entrada una transmisión de vídeo de Kinesis (`Input`), información sobre el bucket de Amazon S3 (`Output`) y un ARN de tema de Amazon SNS (`StreamProcessorNotificationChannel`). También puede proporcionar un ID de clave de KMS para cifrar los datos que se envían al bucket de S3. Especifica lo quiere detectar en `Settings`, como personas, paquetes y personas, o mascotas, personas y paquetes. También puede especificar en qué parte del fotograma quiere que Amazon Rekognition supervise con `RegionsOfInterest`. A continuación, se muestra un ejemplo de JSON para la solicitud `CreateStreamProcessor`.

```
{
  "DataSharingPreference": { "OptIn":TRUE
  },
  "Input": {
    "KinesisVideoStream": {
      "Arn": "arn:aws:kinesisvideo:us-east-1:nnnnnnnnnnnn:stream/muh_video_stream/
nnnnnnnnnnnn"
    }
  },
  "KmsKeyId": "muhkey",
  "Name": "muh-default_stream_processor",
  "Output": {
    "S3Destination": {
      "Bucket": "s3bucket",
      "KeyPrefix": "s3prefix"
    }
  },
  "NotificationChannel": {
    "SNSTopicArn": "arn:aws:sns:us-east-2:nnnnnnnnnnnn:MyTopic"
  },
  "RoleArn": "arn:aws:iam::nnnnnnnnn:role/Admin",
  "Settings": {
    "ConnectedHome": {
      "Labels": [
```

```
        "PET"
      ]
    "MinConfidence": 80
    }
  },
  "RegionsOfInterest": [
    {
      "BoundingBox": {
        "Top": 0.11,
        "Left": 0.22,
        "Width": 0.33,
        "Height": 0.44
      }
    },
    {
      "Polygon": [
        {
          "X": 0.11,
          "Y": 0.11
        },
        {
          "X": 0.22,
          "Y": 0.22
        },
        {
          "X": 0.33,
          "Y": 0.33
        }
      ]
    }
  ]
}
```

Tenga en cuenta que puede cambiar el valor `MinConfidence` cuando especifique el valor `ConnectedHomeSettings` para el procesador de transmisiones. `MinConfidence` es un valor numérico que va de 0 a 100 y que indica la certeza del algoritmo con respecto a sus predicciones. Por ejemplo, una notificación para `person` con un valor de confianza de 90 significa que el algoritmo está absolutamente seguro de que la persona está presente en el vídeo. Un valor de confianza de 10 indica que puede haber una persona. Puede establecer `MinConfidence` en el valor que desee entre 0 y 100, en función de la frecuencia con la que desee recibir notificaciones. Por ejemplo, si

quiere recibir notificaciones solo cuando Rekognition esté absolutamente seguro de que hay un paquete en el fotograma del vídeo, puede configurar `MinConfidence` en 90.

De forma predeterminada, `MinConfidence` está establecido en 50. Si quiere optimizar el algoritmo para obtener una mayor precisión, puede configurar `MinConfidence` para que sea superior a 50. De este modo, recibirá menos notificaciones, pero cada notificación será más fiable. Si quiere optimizar el algoritmo para que sea más exhaustivo, puede configurar `MinConfidence` para que sea inferior a 50 para recibir más notificaciones.

Iniciar el procesador de flujo de detección de etiquetas de Amazon Rekognition Video

Se comienza a analizar el vídeo en streaming llamando a StartStreamProcessor con el nombre del procesador de streaming que especificó en `CreateStreamProcessor`. Cuando ejecuta la operación `StartStreamProcessor` en un procesador de flujo de detección de etiquetas, ingresa la información de inicio y finalización para determinar el tiempo de procesamiento.

Al iniciar el procesador de flujo, el estado del procesador de flujo de detección de etiquetas cambia de las siguientes maneras:

1. Cuando llama a `StartStreamProcessor`, el estado del procesador de flujo de detección de etiquetas pasa de `STOPPED` o `FAILED` a `STARTING`.
2. Mientras el procesador de flujo de detección de etiquetas está en funcionamiento, permanece en `STARTING`.
3. Cuando el procesador de flujo de detección de etiquetas termina de funcionar, el estado pasa a ser `STOPPED` o `FAILED`.

`StartSelector` especifica el punto de partida de la transmisión de Kinesis para iniciar el procesamiento. Puede utilizar la marca de tiempo del productor de KVS o el número de fragmento de KVS. Para obtener más información, consulte Fragment.

> **Note**
>
> Si utiliza la marca de tiempo de KVS Producer, debe introducir la hora en milisegundos.

`StopSelector` especifica cuándo dejar de procesar la transmisión. Puede especificar una cantidad máxima de tiempo para procesar el vídeo. El valor predeterminado es una duración máxima de 10

segundos. Tenga en cuenta que el tiempo de procesamiento real puede ser un poco más largo que la duración máxima, en función del tamaño de los fragmentos de KVS individuales. Si se alcanza o se supera la duración máxima al final de un fragmento, el tiempo de procesamiento se detiene.

A continuación, se muestra un ejemplo de JSON para la solicitud `StartStreamProcessor`.

```
{
   "Name": "string",
   "StartSelector": {
     "KVSStreamStartSelector": {
         "KVSProducerTimestamp": 1655930623123
      },
        "StopSelector": {
            "MaxDurationInSeconds": 11
      }
   }
}
```

Si el procesador de streaming comienza correctamente, se devuelve una respuesta HTTP 200. Se incluye un cuerpo JSON vacío.

Analizar los resultados de la detección de etiquetas

Amazon Rekognition Video publica las notificaciones de un procesador de flujos de detección de etiquetas de tres maneras: notificaciones de Amazon SNS para eventos de detección de objetos, una notificación de Amazon SNS para un resumen end-of-session y un informe detallado de bucket de Amazon S3.

- Notificaciones de Amazon SNS para eventos de detección de objetos.

  Si se detectan etiquetas en la transmisión de vídeo, recibirá notificaciones de Amazon SNS sobre eventos de detección de objetos. Amazon Rekognition publica una notificación la primera vez que se detecta un objeto de interés en la secuencia de vídeo. Las notificaciones incluyen información como el tipo de etiqueta detectada, la confianza y un enlace a la imagen principal. También incluyen una imagen recortada de la persona o el objeto detectado y una marca de tiempo de detección. La notificación de SNS tiene el siguiente formato:

```
{"Subject": "Rekognition Stream Processing Event",
    "Message": {
        "inputInformation": {
            "kinesisVideo": {
                "streamArn": string
            }
        },
        "eventNamespace": {
            "type": "LABEL_DETECTED"
        },
        "labels": [{
            "id": string,
            "name": "PERSON" | "PET" | "PACKAGE",
            "frameImageUri": string,
            "croppedImageUri": string,
            "videoMapping": {
                "kinesisVideoMapping": {
                    "fragmentNumber": string,
                    "serverTimestamp": number,
                    "producerTimestamp": number,
                    "frameOffsetMillis": number
                }
            },
            "boundingBox": {
                "left": number,
                "top": number,
                "height": number,
                "width": number
            }
        }],
        "eventId": string,
        "tags": {
            [string]: string
        },
        "sessionId": string,
        "startStreamProcessorRequest": object
    }
}
```

- Resumen de Amazon SNS. end-of-session

  También recibirá una notificación de Amazon SNS cuando finaliza la sesión de procesamiento de secuencias. En esta notificación se muestran los metadatos de la sesión. Esto incluye detalles

como la duración de la transmisión que se procesó. La notificación de SNS tiene el siguiente formato:

```
{"Subject": "Rekognition Stream Processing Event",
    "Message": {
        "inputInformation": {
            "kinesisVideo": {
                "streamArn": string,
                "processedVideoDurationMillis": number
            }
        },
        "eventNamespace": {
            "type": "STREAM_PROCESSING_COMPLETE"
        },
        "streamProcessingResults": {
            "message": string
        },
        "eventId": string,
        "tags": {
            [string]: string
        },
        "sessionId": string,
        "startStreamProcessorRequest": object
    }
}
```

- Informe de bucket de Amazon S3.

  Amazon Rekognition Video publica los resultados de inferencia detallados de una operación de análisis de vídeo en el bucket de Amazon S3 que se proporciona en la operación `CreateStreamProcessor`. Estos resultados incluyen fotogramas de imágenes en los que se detectó por primera vez un objeto de interés o una persona.

  Los marcos están disponibles en S3 en la siguiente ruta: ObjectKeyPrefix// StreamProcessorNameSessionId/*`service_determined_unique_path`*. En esta ruta, LabelKeyPrefixes un argumento opcional proporcionado por el cliente, StreamProcessorNamees el nombre del recurso del procesador de transmisión y SessionIdes un identificador único para la sesión de procesamiento de transmisión. Sustitúyalos según su situación.

Detección de etiquetas personalizadas

Etiquetas personalizadas de Amazon Rekognition permite identificar los objetos y las escenas de las imágenes específicas de sus necesidades empresariales, como logotipos o piezas de máquinas de ingeniería. Para más información, consulte What is Amazon Rekognition Custom Labels? en la Guía para desarrolladores de Etiquetas personalizadas de Amazon Rekognition.

Detección y análisis de rostros

Amazon Rekognition APIs le proporciona una herramienta que puede utilizar para detectar y analizar rostros en imágenes y vídeos. Esta sección proporciona información general sobre las operaciones sin almacenamiento para el análisis de rostros. Entre estas se incluyen funcionalidades como la detección de referencias faciales, el análisis de emociones y la comparación de rostros.

Amazon Rekognition puede identificar referencias faciales (por ejemplo, la posición de los ojos), detectar emociones (por ejemplo, felicidad o tristeza) y otros atributos (por ejemplo, la presencia de gafas o el ocultamiento facial). Cuando se detecta un rostro, el sistema analiza los atributos faciales y devuelve una puntuación de confianza para cada atributo.



Esta sección contiene ejemplos de operaciones de imágenes y vídeo.

Para obtener más información acerca de cómo usar las operaciones de imágenes de Rekognition, consulte Trabajar con imágenes.

Para obtener más información acerca de cómo usar las operaciones de vídeo de Rekognition, consulte Cómo trabajar con operaciones de análisis de vídeos almacenados.

Tenga en cuenta que estas operaciones son sin almacenamiento. Puede utilizar las operaciones de almacenamiento y las colecciones de rostros para guardar los metadatos faciales de los rostros detectados en una imagen. Posteriormente puede buscar los rostros almacenados en imágenes y vídeos. Por ejemplo, esto le permite buscar una persona específica en un vídeo. Para obtener más información, consulte Búsqueda de rostros en una colección.

Para obtener más información, consulte la sección Rostros de Amazon Rekognition. FAQs

> **Note**
>
> Los modelos de detección de rostros que utilizan Amazon Rekognition Image y Amazon Rekognition Video no admiten la detección de rostros en personajes o entidades no humanas. cartoon/animated Si quiere detectar personajes de dibujos animados en imágenes o vídeos, le recomendamos que utilice Etiquetas personalizadas de Amazon Rekognition. Para más información, consulte la Guía para desarrolladores de Etiquetas personalizadas de Amazon Rekognition.

Temas

- Información general sobre la detección y la comparación de rostros
- Directrices sobre los atributos faciales
- Detección de rostros en una imagen
- Comparación de rostros en imágenes
- Detección de rostros en un vídeo almacenado

Información general sobre la detección y la comparación de rostros

Amazon Rekognition proporciona acceso a los usuarios a dos aplicaciones principales de machine learning para imágenes que contienen rostros: detección de rostros y comparación de rostros. Posibilitan características cruciales, como el análisis facial y la verificación de identidad, por lo que son imprescindibles para diversas aplicaciones, desde la seguridad hasta la organización de fotografías personales.

Detección de rostros

Los sistemas de detección de rostros abordan la pregunta: “¿aparece algún rostro en esta imagen?” al usarlos. Entre los aspectos clave de la detección de rostros se incluyen los siguientes:

- Ubicación y orientación: determina la presencia, ubicación, escala y orientación de los rostros en las imágenes o los fotogramas de vídeo.
- Atributos faciales: detecta los rostros con independencia de atributos como el género, la edad o el vello facial.
- Información adicional: proporciona detalles sobre el ocultamiento facial y la dirección de la mirada.

Comparación de rostros

Los sistemas de comparación de rostros se centran en la pregunta: “¿coincide el rostro de una imagen con un rostro de otra imagen?” al usarlos. Entre las funcionalidades del sistema de comparación de rostros se incluyen las siguientes:

- Predicciones de coincidencia de rostros: compara un rostro de una imagen con un rostro de una base de datos proporcionada para predecir las coincidencias.
- Manejo de atributos faciales: maneja los atributos para comparar rostros con independencia de la expresión, el vello facial y la edad.

Puntuaciones de confianza y detecciones perdidas

Tanto los sistemas de detección como de comparación de rostros utilizan puntuaciones de confianza. Una puntuación de confianza indica la probabilidad de las predicciones, como la presencia de un rostro o una coincidencia entre rostros. Las puntuaciones más altas indican mayor probabilidad. Por ejemplo, un 90 % de confianza sugiere una probabilidad de detección o coincidencia correcta superior al 60 %.

Si un sistema de detección de rostros no detecta correctamente un rostro o proporciona una predicción de baja fiabilidad para un rostro real, se omite el dato negativo. detection/false Si el sistema predice incorrectamente la presencia de un rostro con un nivel de confianza alto, se trata de un falso alarm/false positivo.

Del mismo modo, un sistema de comparación facial puede no hacer coincidir dos rostros que pertenezcan a la misma persona (detection/false negativo omitido) o puede predecir erróneamente que dos rostros de personas diferentes son de la misma persona (falso alarm/false positivo).

Diseño de aplicaciones y establecimiento de umbrales

- Puede establecer un umbral que especifique el nivel de confianza mínimo necesario para devolver un resultado. Elegir los umbrales de confianza adecuados es esencial para el diseño de las aplicaciones y la toma de decisiones en función de los resultados del sistema.
- El nivel de confianza que elija debe reflejar su caso de uso. A continuación, se indican algunos ejemplos de casos de uso y umbrales de confianza:
- • Aplicaciones fotográficas: un umbral más bajo (por ejemplo, el 80 %) puede ser suficiente para identificar a los miembros de una familia en las fotos.
  - Escenarios de alto riesgo: en los casos de uso en los que el riesgo de detección perdida o de falsa alarma es mayor, como en las aplicaciones de seguridad, el sistema debe utilizar un nivel de confianza más alto. En estos casos, se recomienda un umbral más alto (por ejemplo, el 99 %) para conseguir coincidencias faciales precisas.

Para obtener más información acerca de cómo establecer y comprender los requisitos de confianza, consulte Búsqueda de rostros en una colección.

Directrices sobre los atributos faciales

A continuación, se incluye información detallada sobre cómo procesa y devuelve los atributos faciales Amazon Rekognition.

- FaceDetail Objeto: por cada rostro detectado, se devuelve un FaceDetail objeto. FaceDetail Contiene datos sobre los puntos de referencia del rostro, la calidad, la postura y más.
- Predicciones de atributos: se predicen atributos como la emoción, el género, la edad y otros. Se asigna un nivel de confianza a cada predicción y las predicciones se devuelven con la puntuación de confianza correspondiente. Se recomienda un umbral de confianza del 99 % para los casos de uso confidenciales. Para la estimación de la edad, el punto medio del rango de edad previsto ofrece la mejor aproximación.

Tenga en cuenta que las predicciones de género y de emociones se basan en la apariencia física y no deben usarse para determinar la identidad de género o el estado emocional real. La predicción binaria del sexo (masculino/femenino) utiliza la apariencia física de un rostro de una imagen determinada. No indica la identidad de género de una persona y no debe utilizarse Rekognition para realizar esta determinación. No es recomendable utilizar predicciones binarias de género para tomar decisiones que podrían afectar a los derechos, la privacidad o el acceso de una persona a los servicios. Del mismo modo, la predicción de emociones no indica el estado emocional interno real

de una persona y no debe utilizarse Rekognition para realizar esta determinación. Una persona que pone cara de felicidad en una imagen puede parecer contenta, si bien puede que no se sienta feliz.

Aplicación y casos de uso

A continuación, se describen algunas aplicaciones y casos de uso prácticos para estos atributos:

- Aplicaciones: atributos como la sonrisa, la postura y la nitidez se pueden utilizar para seleccionar imágenes de perfil o realizar estimaciones demográficas de forma anónima.
- Casos de uso comunes: las aplicaciones de redes sociales y la estimación demográfica en eventos o tiendas minoristas son ejemplos típicos.

Para obtener información más detallada sobre cada atributo, consulte FaceDetail.

Detección de rostros en una imagen

Amazon Rekognition Image DetectFacesproporciona la operación que busca rasgos faciales clave, como los ojos, la nariz y la boca, para detectar rostros en una imagen de entrada. Amazon Rekognition Image detecta las 100 caras más grandes de una imagen.

Puede proporcionar la imagen de entrada como una matriz de bytes de imagen (con codificación en base64) o especificar un objeto de Amazon S3. En este procedimiento subirá una imagen (JPEG o PNG) en su bucket de S3; y especificará el nombre de clave del objeto.

Detección de rostros en una imagen

1. Si aún no lo ha hecho:

   a. Cree o actualice un usuario con los permisos `AmazonRekognitionFullAccess` y `AmazonS3ReadOnlyAccess`. Para obtener más información, consulte Paso 1: Configurar una cuenta de AWS y crear un usuario.

   b. Instale y configure el y el AWS CLI . AWS SDKs Para obtener más información, consulte Paso 2: Configure y AWS CLI AWS SDKs.

2. Suba una imagen (que contenga uno o varios rostros) en el bucket de S3.

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

3. Utilice los siguientes ejemplos para llamar a `DetectFaces`.

Java

Este ejemplo muestra el rango de edades estimado de los rostros detectados y muestra el código JSON para todos los atributos faciales detectados. Cambie el valor de `photo` por el nombre de archivo de la imagen. Cambie el valor de `amzn-s3-demo-bucket` por el bucket de Amazon S3 donde se almacena la imagen.

```
//Copyright 2018 Amazon.com, Inc. or its affiliates. All Rights Reserved.
//PDX-License-Identifier: MIT-0 (For details, see https://github.com/awsdocs/
amazon-rekognition-developer-guide/blob/master/LICENSE-SAMPLECODE.)

package aws.example.rekognition.image;

import com.amazonaws.services.rekognition.AmazonRekognition;
import com.amazonaws.services.rekognition.AmazonRekognitionClientBuilder;
import com.amazonaws.services.rekognition.model.AmazonRekognitionException;
import com.amazonaws.services.rekognition.model.Image;
import com.amazonaws.services.rekognition.model.S3Object;
import com.amazonaws.services.rekognition.model.AgeRange;
import com.amazonaws.services.rekognition.model.Attribute;
import com.amazonaws.services.rekognition.model.DetectFacesRequest;
import com.amazonaws.services.rekognition.model.DetectFacesResult;
import com.amazonaws.services.rekognition.model.FaceDetail;
import com.fasterxml.jackson.databind.ObjectMapper;
import java.util.List;


public class DetectFaces {


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

      String photo = "input.jpg";
      String bucket = "bucket";

      AmazonRekognition rekognitionClient =
 AmazonRekognitionClientBuilder.defaultClient();


      DetectFacesRequest request = new DetectFacesRequest()
         .withImage(new Image()
            .withS3Object(new S3Object()
```

```
                .withName(photo)
                .withBucket(bucket)))
           .withAttributes(Attribute.ALL);
        // Replace Attribute.ALL with Attribute.DEFAULT to get default values.

        try {
           DetectFacesResult result = rekognitionClient.detectFaces(request);
           List < FaceDetail > faceDetails = result.getFaceDetails();

           for (FaceDetail face: faceDetails) {
              if (request.getAttributes().contains("ALL")) {
                 AgeRange ageRange = face.getAgeRange();
                 System.out.println("The detected face is estimated to be between
 "
                    + ageRange.getLow().toString() + " and " +
 ageRange.getHigh().toString()
                    + " years old.");
                 System.out.println("Here's the complete set of attributes:");
              } else { // non-default attributes have null values.
                 System.out.println("Here's the default set of attributes:");
              }

              ObjectMapper objectMapper = new ObjectMapper();

 System.out.println(objectMapper.writerWithDefaultPrettyPrinter().writeValueAsString(fac
           }

        } catch (AmazonRekognitionException e) {
           e.printStackTrace();
        }

     }

}
```

Java V2

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

```
import java.util.List;
```

```
//snippet-start:[rekognition.java2.detect_labels.import]
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import software.amazon.awssdk.services.rekognition.model.S3Object;
import software.amazon.awssdk.services.rekognition.model.DetectFacesRequest;
import software.amazon.awssdk.services.rekognition.model.DetectFacesResponse;
import software.amazon.awssdk.services.rekognition.model.Image;
import software.amazon.awssdk.services.rekognition.model.Attribute;
import software.amazon.awssdk.services.rekognition.model.FaceDetail;
import software.amazon.awssdk.services.rekognition.model.AgeRange;

//snippet-end:[rekognition.java2.detect_labels.import]

public class DetectFaces {

    public static void main(String[] args) {
        final String usage = "\n" +
            "Usage: " +
            "   <bucket> <image>\n\n" +
            "Where:\n" +
            "   bucket - The name of the Amazon S3 bucket that contains the
 image (for example, ,amzn-s3-demo-bucket)." +
            "   image - The name of the image located in the Amazon S3 bucket
 (for example, Lake.png). \n\n";

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucket = args[0];
        String image = args[1];
        Region region = Region.US_WEST_2;
        RekognitionClient rekClient = RekognitionClient.builder()
            .region(region)
            .credentialsProvider(ProfileCredentialsProvider.create("profile-
name"))
            .build();

        getLabelsfromImage(rekClient, bucket, image);
        rekClient.close();
    }
```

```
   // snippet-start:[rekognition.java2.detect_labels_s3.main]
   public static void getLabelsfromImage(RekognitionClient rekClient, String
 bucket, String image) {

       try {
           S3Object s3Object = S3Object.builder()
               .bucket(bucket)
               .name(image)
               .build() ;

           Image myImage = Image.builder()
               .s3Object(s3Object)
               .build();

           DetectFacesRequest facesRequest = DetectFacesRequest.builder()
                   .attributes(Attribute.ALL)
                   .image(myImage)
                   .build();

               DetectFacesResponse facesResponse =
 rekClient.detectFaces(facesRequest);
               List<FaceDetail> faceDetails = facesResponse.faceDetails();
               for (FaceDetail face : faceDetails) {
                   AgeRange ageRange = face.ageRange();
                   System.out.println("The detected face is estimated to be
 between "
                                   + ageRange.low().toString() + " and " +
 ageRange.high().toString()
                                   + " years old.");

                   System.out.println("There is a smile :
 "+face.smile().value().toString());
               }

       } catch (RekognitionException e) {
           System.out.println(e.getMessage());
           System.exit(1);
       }
   }
 // snippet-end:[rekognition.java2.detect_labels.main]
}
```

AWS CLI

En este ejemplo, se muestra el resultado JSON de la `detect-faces` AWS CLI operación. Reemplace `file` por el nombre de un archivo de imagen. Sustituya `amzn-s3-demo-bucket` por el nombre del bucket de Amazon S3; que contiene el archivo de imagen.

```
aws rekognition detect-faces --image "{"S3Object":{"Bucket":"amzn-s3-demo-
bucket,"Name":"image-name"}}"\
                            --attributes "ALL" --profile profile-name --region
 region-name
```

Si accede a la CLI en un dispositivo Windows, utilice comillas dobles en lugar de comillas simples y evite las comillas dobles internas con una barra invertida (es decir, \) para corregir cualquier error del analizador que pueda encontrar. Por ver un ejemplo, consulte lo siguiente:

```
aws rekognition detect-faces --image "{\"S3Object\":{\"Bucket\":\"amzn-s3-demo-
bucket\",\"Name\":\"image-name\"}}" --attributes "ALL"
--profile profile-name --region region-name
```

Python

Este ejemplo muestra el rango de edades estimado y otros atributos de los rostros detectados y muestra el código JSON para todos los atributos faciales detectados. Cambie el valor de `photo` por el nombre de archivo de la imagen. Cambie el valor de `amzn-s3-demo-bucket` por el bucket de Amazon S3 donde se almacena la imagen. Sustituya el valor de `profile_name` en la línea que crea la sesión de Rekognition por el nombre de su perfil de desarrollador.

```
import boto3
import json

def detect_faces(photo, bucket, region):

    session = boto3.Session(profile_name='profile-name',
                            region_name=region)
    client = session.client('rekognition', region_name=region)
```

```
    response = client.detect_faces(Image={'S3Object':
{'Bucket':bucket,'Name':photo}},
                                   Attributes=['ALL'])

    print('Detected faces for ' + photo)
    for faceDetail in response['FaceDetails']:
        print('The detected face is between ' + str(faceDetail['AgeRange']
['Low'])
              + ' and ' + str(faceDetail['AgeRange']['High']) + ' years old')

        print('Here are the other attributes:')
        print(json.dumps(faceDetail, indent=4, sort_keys=True))

        # Access predictions for individual face details and print them
        print("Gender: " + str(faceDetail['Gender']))
        print("Smile: " + str(faceDetail['Smile']))
        print("Eyeglasses: " + str(faceDetail['Eyeglasses']))
        print("Face Occluded: " + str(faceDetail['FaceOccluded']))
        print("Emotions: " + str(faceDetail['Emotions'][0]))

    return len(response['FaceDetails'])

def main():
    photo='photo'
    bucket='amzn-s3-demo-bucket'
    region='region'
    face_count=detect_faces(photo, bucket, region)
    print("Faces detected: " + str(face_count))

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

.NET

Este ejemplo muestra el rango de edades estimado de los rostros detectados y muestra el código JSON para todos los atributos faciales detectados. Cambie el valor de `photo` por el nombre de archivo de la imagen. Cambie el valor de `amzn-s3-demo-bucket` por el bucket de Amazon S3 donde se almacena la imagen.

```
//Copyright 2018 Amazon.com, Inc. or its affiliates. All Rights Reserved.
//PDX-License-Identifier: MIT-0 (For details, see https://github.com/awsdocs/
amazon-rekognition-developer-guide/blob/master/LICENSE-SAMPLECODE.)
```

```
using System;
using System.Collections.Generic;
using Amazon.Rekognition;
using Amazon.Rekognition.Model;

public class DetectFaces
{
    public static void Example()
    {
        String photo = "input.jpg";
        String bucket = "amzn-s3-demo-bucket";

        AmazonRekognitionClient rekognitionClient = new
 AmazonRekognitionClient();

        DetectFacesRequest detectFacesRequest = new DetectFacesRequest()
        {
            Image = new Image()
            {
                S3Object = new S3Object()
                {
                    Name = photo,
                    Bucket = bucket
                },
            },
            // Attributes can be "ALL" or "DEFAULT".
            // "DEFAULT": BoundingBox, Confidence, Landmarks, Pose, and Quality.
            // "ALL": See https://docs.aws.amazon.com/sdkfornet/v3/apidocs/
items/Rekognition/TFaceDetail.html
            Attributes = new List<String>() { "ALL" }
        };

        try
        {
            DetectFacesResponse detectFacesResponse =
 rekognitionClient.DetectFaces(detectFacesRequest);
            bool hasAll = detectFacesRequest.Attributes.Contains("ALL");
            foreach(FaceDetail face in detectFacesResponse.FaceDetails)
            {
                Console.WriteLine("BoundingBox: top={0} left={1} width={2}
 height={3}", face.BoundingBox.Left,
                    face.BoundingBox.Top, face.BoundingBox.Width,
 face.BoundingBox.Height);
```

```
                Console.WriteLine("Confidence: {0}\nLandmarks: {1}\nPose:
 pitch={2} roll={3} yaw={4}\nQuality: {5}",
                    face.Confidence, face.Landmarks.Count, face.Pose.Pitch,
                    face.Pose.Roll, face.Pose.Yaw, face.Quality);
                if (hasAll)
                    Console.WriteLine("The detected face is estimated to be
 between " +
                        face.AgeRange.Low + " and " + face.AgeRange.High + "
 years old.");
            }
        }
        catch (Exception e)
        {
            Console.WriteLine(e.Message);
        }
    }
}
```

Ruby

En este ejemplo, se muestra el rango de edad estimado de los rostros detectados y se enumeran varios atributos faciales. Cambie el valor de `photo` por el nombre de archivo de la imagen. Cambie el valor de `amzn-s3-demo-bucket` por el bucket de Amazon S3 donde se almacena la imagen.

```
  # Add to your Gemfile
  # gem 'aws-sdk-rekognition'
  require 'aws-sdk-rekognition'
  credentials = Aws::Credentials.new(
     ENV['AWS_ACCESS_KEY_ID'],
     ENV['AWS_SECRET_ACCESS_KEY']
  )
  bucket = 'bucket' # the bucketname without s3://
  photo  = 'input.jpg'# the name of file
  client   = Aws::Rekognition::Client.new credentials: credentials
  attrs = {
    image: {
      s3_object: {
        bucket: bucket,
        name: photo
      },
```

```
    },
    attributes: ['ALL']
  }
  response = client.detect_faces attrs
  puts "Detected faces for: #{photo}"
  response.face_details.each do |face_detail|
    low  = face_detail.age_range.low
    high = face_detail.age_range.high
    puts "The detected face is between: #{low} and #{high} years old"
    puts "All other attributes:"
    puts "  bounding_box.width:     #{face_detail.bounding_box.width}"
    puts "  bounding_box.height:    #{face_detail.bounding_box.height}"
    puts "  bounding_box.left:      #{face_detail.bounding_box.left}"
    puts "  bounding_box.top:       #{face_detail.bounding_box.top}"
    puts "  age.range.low:          #{face_detail.age_range.low}"
    puts "  age.range.high:         #{face_detail.age_range.high}"
    puts "  smile.value:            #{face_detail.smile.value}"
    puts "  smile.confidence:       #{face_detail.smile.confidence}"
    puts "  eyeglasses.value:       #{face_detail.eyeglasses.value}"
    puts "  eyeglasses.confidence:  #{face_detail.eyeglasses.confidence}"
    puts "  sunglasses.value:       #{face_detail.sunglasses.value}"
    puts "  sunglasses.confidence:  #{face_detail.sunglasses.confidence}"
    puts "  gender.value:           #{face_detail.gender.value}"
    puts "  gender.confidence:      #{face_detail.gender.confidence}"
    puts "  beard.value:            #{face_detail.beard.value}"
    puts "  beard.confidence:       #{face_detail.beard.confidence}"
    puts "  mustache.value:         #{face_detail.mustache.value}"
    puts "  mustache.confidence:    #{face_detail.mustache.confidence}"
    puts "  eyes_open.value:        #{face_detail.eyes_open.value}"
    puts "  eyes_open.confidence:   #{face_detail.eyes_open.confidence}"
    puts "  mout_open.value:        #{face_detail.mouth_open.value}"
    puts "  mout_open.confidence:   #{face_detail.mouth_open.confidence}"
    puts "  emotions[0].type:       #{face_detail.emotions[0].type}"
    puts "  emotions[0].confidence: #{face_detail.emotions[0].confidence}"
    puts "  landmarks[0].type:      #{face_detail.landmarks[0].type}"
    puts "  landmarks[0].x:         #{face_detail.landmarks[0].x}"
    puts "  landmarks[0].y:         #{face_detail.landmarks[0].y}"
    puts "  pose.roll:              #{face_detail.pose.roll}"
    puts "  pose.yaw:               #{face_detail.pose.yaw}"
    puts "  pose.pitch:             #{face_detail.pose.pitch}"
    puts "  quality.brightness:     #{face_detail.quality.brightness}"
    puts "  quality.sharpness:      #{face_detail.quality.sharpness}"
    puts "  confidence:             #{face_detail.confidence}"
    puts "------------"
```

```
    puts ""
  end
```

Node.js

En este ejemplo, se muestra el rango de edad estimado de los rostros detectados y se enumeran varios atributos faciales. Cambie el valor de `photo` por el nombre de archivo de la imagen. Cambie el valor de `amzn-s3-demo-bucket` por el bucket de Amazon S3 donde se almacena la imagen.

Sustituya el valor de `profile_name` en la línea que crea la sesión de Rekognition por el nombre de su perfil de desarrollador.

Si utiliza TypeScript definiciones, puede que necesite utilizarlas en `import AWS from 'aws-sdk'` lugar de`const AWS = require('aws-sdk')`, para ejecutar el programa con Node.js. Puede consultar el AWS SDK para Javascript si quiere obtener más información. En función de cómo tenga establecidas las opciones de configuración, es posible que también tenga que especificar su región con `AWS.config.update({region:`*`region`*`});`.

```
// Load the SDK
var AWS = require('aws-sdk');
const bucket = 'bucket-name' // the bucketname without s3://
const photo  = 'photo-name' // the name of file

var credentials = new AWS.SharedIniFileCredentials({profile: 'profile-name'});
AWS.config.credentials = credentials;
AWS.config.update({region:'region-name'});

const client = new AWS.Rekognition();
const params = {
  Image: {
    S3Object: {
      Bucket: bucket,
      Name: photo
    },
  },
  Attributes: ['ALL']
}

client.detectFaces(params, function(err, response) {
```

```
    if (err) {
      console.log(err, err.stack); // an error occurred
    } else {
      console.log(`Detected faces for: ${photo}`)
      response.FaceDetails.forEach(data => {
        let low  = data.AgeRange.Low
        let high = data.AgeRange.High
        console.log(`The detected face is between: ${low} and ${high} years
old`)
        console.log("All other attributes:")
        console.log(`  BoundingBox.Width:      ${data.BoundingBox.Width}`)
        console.log(`  BoundingBox.Height:     ${data.BoundingBox.Height}`)
        console.log(`  BoundingBox.Left:       ${data.BoundingBox.Left}`)
        console.log(`  BoundingBox.Top:        ${data.BoundingBox.Top}`)
        console.log(`  Age.Range.Low:          ${data.AgeRange.Low}`)
        console.log(`  Age.Range.High:         ${data.AgeRange.High}`)
        console.log(`  Smile.Value:            ${data.Smile.Value}`)
        console.log(`  Smile.Confidence:       ${data.Smile.Confidence}`)
        console.log(`  Eyeglasses.Value:       ${data.Eyeglasses.Value}`)
        console.log(`  Eyeglasses.Confidence:  ${data.Eyeglasses.Confidence}`)
        console.log(`  Sunglasses.Value:       ${data.Sunglasses.Value}`)
        console.log(`  Sunglasses.Confidence:  ${data.Sunglasses.Confidence}`)
        console.log(`  Gender.Value:           ${data.Gender.Value}`)
        console.log(`  Gender.Confidence:      ${data.Gender.Confidence}`)
        console.log(`  Beard.Value:            ${data.Beard.Value}`)
        console.log(`  Beard.Confidence:       ${data.Beard.Confidence}`)
        console.log(`  Mustache.Value:         ${data.Mustache.Value}`)
        console.log(`  Mustache.Confidence:    ${data.Mustache.Confidence}`)
        console.log(`  EyesOpen.Value:         ${data.EyesOpen.Value}`)
        console.log(`  EyesOpen.Confidence:    ${data.EyesOpen.Confidence}`)
        console.log(`  MouthOpen.Value:        ${data.MouthOpen.Value}`)
        console.log(`  MouthOpen.Confidence:   ${data.MouthOpen.Confidence}`)
        console.log(`  Emotions[0].Type:       ${data.Emotions[0].Type}`)
        console.log(`  Emotions[0].Confidence: ${data.Emotions[0].Confidence}`)
        console.log(`  Landmarks[0].Type:      ${data.Landmarks[0].Type}`)
        console.log(`  Landmarks[0].X:         ${data.Landmarks[0].X}`)
        console.log(`  Landmarks[0].Y:         ${data.Landmarks[0].Y}`)
        console.log(`  Pose.Roll:              ${data.Pose.Roll}`)
        console.log(`  Pose.Yaw:               ${data.Pose.Yaw}`)
        console.log(`  Pose.Pitch:             ${data.Pose.Pitch}`)
        console.log(`  Quality.Brightness:     ${data.Quality.Brightness}`)
        console.log(`  Quality.Sharpness:      ${data.Quality.Sharpness}`)
        console.log(`  Confidence:             ${data.Confidence}`)
        console.log("------------")
```

```
        console.log("")
      }) // for response.faceDetails
    } // if
  });
```

DetectFaces solicitud de operación

La entrada de `DetectFaces` es una imagen. En este ejemplo, la imagen se carga desde un bucket de Amazon S3. El parámetro `Attributes` especifica que se deben devolver todos los atributos faciales. Para obtener más información, consulte Trabajar con imágenes.

```
{
    "Image": {
        "S3Object": {
            "Bucket": "amzn-s3-demo-bucket",
            "Name": "input.jpg"
        }
    },
    "Attributes": [
        "ALL"
    ]
}
```

DetectFaces respuesta de operación

`DetectFaces` devuelve la siguiente información por cada rostro detectado:

- Cuadro delimitador: las coordenadas del cuadro delimitador que rodea el rostro.
- Confianza: grado de confianza de que el cuadro delimitador contiene un rostro.
- Referencias faciales: una matriz de referencias faciales. Para cada referencia (como el ojo izquierdo, el ojo derecho y la boca), la respuesta proporciona las coordenadas x, y.
- Atributos faciales: conjunto de atributos faciales, como si el rostro está oculto, que se devuelve como un objeto `FaceDetail`. El set incluye: barba AgeRange, emociones EyeDirection, anteojos, género, bigote EyesOpen FaceOccluded, sonrisa MouthOpen y gafas de sol. Para cada atributo, la respuesta proporciona un valor. El valor puede ser de diferentes tipos, como un tipo booleano (si la persona lleva gafas de sol), una cadena (si la persona es hombre o mujer) o un valor de grado angular (para pitch/yaw la dirección de la mirada). Además, para la mayoría de los atributos la respuesta proporciona también una confianza en el valor detectado para el atributo.

Tenga en cuenta que, si bien EyeDirection los atributos FaceOccluded y se admiten cuando se utilizan`DetectFaces`, no se admiten cuando se analizan vídeos con `StartFaceDetection` y. `GetFaceDetection`

- Calidad: describe el brillo y la nitidez del rostro. Para obtener información acerca de cómo optimizar la detección de rostros, consulte Recomendaciones para la comparación de rostros en las imágenes de entrada.
- Postura: describe la rotación del rostro dentro de la imagen.

La solicitud puede incluir una serie de atributos faciales que quiera que se devuelvan. Siempre se mostrará un subconjunto `DEFAULT` de atributos faciales: `BoundingBox`, `Confidence`, `Pose`, `Quality` y `Landmarks`. Puede solicitar la devolución de atributos faciales específicos (además de la lista predeterminada), utilizando `["DEFAULT", "FACE_OCCLUDED", "EYE_DIRECTION"]` o solo un atributo, por ejemplo `["FACE_OCCLUDED"]`. Puede solicitar todos los atributos faciales usando `["ALL"]`. Solicitar más atributos puede aumentar el tiempo de respuesta.

A continuación se muestra un ejemplo de respuesta de una llamada a la API `DetectFaces`:


```
{
  "FaceDetails": [
    {
      "BoundingBox": {
        "Width": 0.7919622659683228,
        "Height": 0.7510867118835449,
        "Left": 0.08881539851427078,
        "Top": 0.151064932346344
      },
      "AgeRange": {
        "Low": 18,
        "High": 26
      },
      "Smile": {
        "Value": false,
        "Confidence": 89.77348327636719
      },
      "Eyeglasses": {
        "Value": true,
        "Confidence": 99.99996948242188
      },
      "Sunglasses": {
        "Value": true,
```

```
        "Confidence": 93.65237426757812
      },
      "Gender": {
        "Value": "Female",
        "Confidence": 99.85968780517578
      },
      "Beard": {
        "Value": false,
        "Confidence": 77.52591705322266
      },
      "Mustache": {
        "Value": false,
        "Confidence": 94.48904418945312
      },
      "EyesOpen": {
        "Value": true,
        "Confidence": 98.57169342041016
      },
      "MouthOpen": {
        "Value": false,
        "Confidence": 74.33953094482422
      },
      "Emotions": [
        {
          "Type": "SAD",
          "Confidence": 65.56403350830078
        },
        {
          "Type": "CONFUSED",
          "Confidence": 31.277774810791016
        },
        {
          "Type": "DISGUSTED",
          "Confidence": 15.553778648376465
        },
        {
          "Type": "ANGRY",
          "Confidence": 8.012762069702148
        },
        {
          "Type": "SURPRISED",
          "Confidence": 7.621500015258789
        },
        {
```

```
          "Type": "FEAR",
          "Confidence": 7.243380546569824
        },
        {
          "Type": "CALM",
          "Confidence": 5.8196024894714355
        },
        {
          "Type": "HAPPY",
          "Confidence": 2.2830512523651123
        }
      ],
      "Landmarks": [
        {
          "Type": "eyeLeft",
          "X": 0.30225440859794617,
          "Y": 0.41018882393836975
        },
        {
          "Type": "eyeRight",
          "X": 0.6439348459243774,
          "Y": 0.40341562032699585
        },
        {
          "Type": "mouthLeft",
          "X": 0.343580037355423,
          "Y": 0.6951127648353577
        },
        {
          "Type": "mouthRight",
          "X": 0.6306480765342712,
          "Y": 0.6898072361946106
        },
        {
          "Type": "nose",
          "X": 0.47164231538772583,
          "Y": 0.5763645172119141
        },
        {
          "Type": "leftEyeBrowLeft",
          "X": 0.1732882857322693,
          "Y": 0.34452149271965027
        },
        {
```

```
          "Type": "leftEyeBrowRight",
          "X": 0.3655243515968323,
          "Y": 0.33231860399246216
        },
        {
          "Type": "leftEyeBrowUp",
          "X": 0.2671719491481781,
          "Y": 0.31669262051582336
        },
        {
          "Type": "rightEyeBrowLeft",
          "X": 0.5613729953765869,
          "Y": 0.32813435792922974
        },
        {
          "Type": "rightEyeBrowRight",
          "X": 0.7665090560913086,
          "Y": 0.3318614959716797
        },
        {
          "Type": "rightEyeBrowUp",
          "X": 0.6612788438796997,
          "Y": 0.3082450032234192
        },
        {
          "Type": "leftEyeLeft",
          "X": 0.2416982799768448,
          "Y": 0.4085965156555176
        },
        {
          "Type": "leftEyeRight",
          "X": 0.36943578720092773,
          "Y": 0.41230902075767517
        },
        {
          "Type": "leftEyeUp",
          "X": 0.29974061250686646,
          "Y": 0.3971870541572571
        },
        {
          "Type": "leftEyeDown",
          "X": 0.30360740423202515,
          "Y": 0.42347756028175354
        },
```

```
        {
          "Type": "rightEyeLeft",
          "X": 0.5755768418312073,
          "Y": 0.4081145226955414
        },
        {
          "Type": "rightEyeRight",
          "X": 0.7050536870956421,
          "Y": 0.39924031496047974
        },
        {
          "Type": "rightEyeUp",
          "X": 0.642906129360199,
          "Y": 0.39026668667793274
        },
        {
          "Type": "rightEyeDown",
          "X": 0.6423097848892212,
          "Y": 0.41669243574142456
        },
        {
          "Type": "noseLeft",
          "X": 0.4122826159000397,
          "Y": 0.5987403392791748
        },
        {
          "Type": "noseRight",
          "X": 0.5394935011863708,
          "Y": 0.5960900187492371
        },
        {
          "Type": "mouthUp",
          "X": 0.478581964969635,
          "Y": 0.6660456657409668
        },
        {
          "Type": "mouthDown",
          "X": 0.483366996049881,
          "Y": 0.7497162818908691
        },
        {
          "Type": "leftPupil",
          "X": 0.30225440859794617,
          "Y": 0.41018882393836975
```

```
        },
        {
          "Type": "rightPupil",
          "X": 0.6439348459243774,
          "Y": 0.40341562032699585
        },
        {
          "Type": "upperJawlineLeft",
          "X": 0.11031254380941391,
          "Y": 0.3980775475502014
        },
        {
          "Type": "midJawlineLeft",
          "X": 0.19301874935626984,
          "Y": 0.7034031748771667
        },
        {
          "Type": "chinBottom",
          "X": 0.4939905107021332,
          "Y": 0.8877836465835571
        },
        {
          "Type": "midJawlineRight",
          "X": 0.7990140914916992,
          "Y": 0.6899225115776062
        },
        {
          "Type": "upperJawlineRight",
          "X": 0.8548634648323059,
          "Y": 0.38160091638565063
        }
      ],
      "Pose": {
        "Roll": -5.83309268951416,
        "Yaw": -2.4244730472564697,
        "Pitch": 2.6216139793395996
      },
      "Quality": {
        "Brightness": 96.16363525390625,
        "Sharpness": 95.51618957519531
      },
      "Confidence": 99.99872589111328,
      "FaceOccluded": {
        "Value": true,
```

```
        "Confidence": 99.99726104736328
      },
      "EyeDirection": {
        "Yaw": 16.299732,
        "Pitch": -6.407457,
        "Confidence": 99.968704
      }
    }
  ],
  "ResponseMetadata": {
    "RequestId": "8bf02607-70b7-4f20-be55-473fe1bba9a2",
    "HTTPStatusCode": 200,
    "HTTPHeaders": {
      "x-amzn-requestid": "8bf02607-70b7-4f20-be55-473fe1bba9a2",
      "content-type": "application/x-amz-json-1.1",
      "content-length": "3409",
      "date": "Wed, 26 Apr 2023 20:18:50 GMT"
    },
    "RetryAttempts": 0
  }
}
```

Tenga en cuenta lo siguiente:

- Los datos de `Pose` describen la rotación del rostro detectado. Puede utilizar la combinación de los datos de `BoundingBox` y `Pose` para dibujar el cuadro delimitador en los rostros mostrados por la aplicación.
- `Quality` describe el brillo y la nitidez del rostro. Tal vez le resulte útil comparar rostros de imágenes para encontrar el mejor.
- La respuesta anterior muestra todos los valores de `landmarks` que el servicio puede detectar: todos los atributos faciales y las emociones. Para obtener todos estos atributos en la respuesta, debe especificar el parámetro `attributes` con el valor `ALL`. De forma predeterminada, la API `DetectFaces` devuelve solo los siguientes cinco atributos faciales: `BoundingBox`, `Confidence`, `Pose`, `Quality` y `landmarks`. Las referencias predeterminadas que se devuelve son: `eyeLeft`, `eyeRight`, `nose`, `mouthLeft` y `mouthRight`.

Comparación de rostros en imágenes

Con Rekognition puedes comparar rostros entre dos imágenes mediante la operación. CompareFaces Esta característica es útil para aplicaciones como la verificación de identidad o la coincidencia de fotos.

CompareFaces compara un rostro de la imagen de origen con cada rostro de la imagen de destino. Las imágenes se pasan a una de CompareFaces las siguientes formas:

- Representación de una imagen codificada en base64
- Objetos de Amazon S3

Detección de rostros frente a comparación de rostros

La comparación de rostros es distinta de la detección de rostros. La detección de rostros (que utiliza DetectFaces) solo identifica la presencia y ubicación de rostros en una imagen o vídeo. Por el contrario, la comparación de rostros implica comparar un rostro que se ha detectado en una imagen de origen con los rostros de una imagen de destino para encontrar coincidencias.

Umbrales de similitud

Utilice el parámetro `similarityThreshold` para definir el nivel de confianza mínimo necesario para que las coincidencias se incluyan en la respuesta. De forma predeterminada, en la respuesta solo se devuelven los rostros con una puntuación de similitud superior o igual al 80 %.

> **Note**
>
> `CompareFaces` utiliza algoritmos de machine learning, basados en probabilidades. Un falso negativo es una predicción incorrecta de que un rostro de la imagen objetivo tiene una puntuación de confianza de similitud baja en comparación con el rostro de la imagen original. Para reducir la probabilidad de que se produzcan falsos negativos, le recomendamos que compare la imagen de destino con varias imágenes de origen. Si piensa utilizar `CompareFaces` para tomar una decisión que afecte a los derechos, la privacidad o el acceso de una persona a los servicios, le recomendamos que transmita el resultado a una persona para que lo revise y lo valide antes de tomar medidas.

Los siguientes ejemplos de código muestran cómo utilizar las CompareFaces operaciones para varios tipos de operaciones AWS SDKs. En el AWS CLI ejemplo, carga dos imágenes JPEG a su bucket de Amazon S3 y especifica el nombre de la clave del objeto. En los demás ejemplos, puede subir dos archivos desde el sistema de archivos local e introducirlos como una matriz de bytes de imagen.

Para comparar rostros

1. Si aún no lo ha hecho:

   a. Cree o actualice un usuario con `AmazonRekognitionFullAccess` permisos `AmazonS3ReadOnlyAccess` (solo de AWS CLI ejemplo). Para obtener más información, consulte Paso 1: Configurar una cuenta de AWS y crear un usuario.

   b. Instale y configure los AWS CLI y los AWS SDKs. Para obtener más información, consulte Paso 2: Configure y AWS CLI AWS SDKs.

2. Utilice el siguiente código de ejemplo para realizar llamadas a la operación `CompareFaces`.

   Java

   Este ejemplo muestra información sobre rostros coincidentes de las imágenes de origen y de destino que se cargan desde el sistema de archivos local.

   Reemplace los valores de `sourceImage` y `targetImage` por la ruta y el nombre de archivo de las imágenes de origen y de destino.

```
//Copyright 2018 Amazon.com, Inc. or its affiliates. All Rights Reserved.
//PDX-License-Identifier: MIT-0 (For details, see https://github.com/awsdocs/
amazon-rekognition-developer-guide/blob/master/LICENSE-SAMPLECODE.)

package aws.example.rekognition.image;
import com.amazonaws.services.rekognition.AmazonRekognition;
import com.amazonaws.services.rekognition.AmazonRekognitionClientBuilder;
import com.amazonaws.services.rekognition.model.Image;
import com.amazonaws.services.rekognition.model.BoundingBox;
import com.amazonaws.services.rekognition.model.CompareFacesMatch;
import com.amazonaws.services.rekognition.model.CompareFacesRequest;
import com.amazonaws.services.rekognition.model.CompareFacesResult;
import com.amazonaws.services.rekognition.model.ComparedFace;
import java.util.List;
import java.io.File;
import java.io.FileInputStream;
```

```
import java.io.InputStream;
import java.nio.ByteBuffer;
import com.amazonaws.util.IOUtils;

public class CompareFaces {

   public static void main(String[] args) throws Exception{
       Float similarityThreshold = 70F;
       String sourceImage = "source.jpg";
       String targetImage = "target.jpg";
       ByteBuffer sourceImageBytes=null;
       ByteBuffer targetImageBytes=null;

       AmazonRekognition rekognitionClient =
 AmazonRekognitionClientBuilder.defaultClient();

       //Load source and target images and create input parameters
       try (InputStream inputStream = new FileInputStream(new
 File(sourceImage))) {
          sourceImageBytes = ByteBuffer.wrap(IOUtils.toByteArray(inputStream));
       }
       catch(Exception e)
       {
           System.out.println("Failed to load source image " + sourceImage);
           System.exit(1);
       }
       try (InputStream inputStream = new FileInputStream(new
 File(targetImage))) {
           targetImageBytes =
 ByteBuffer.wrap(IOUtils.toByteArray(inputStream));
       }
       catch(Exception e)
       {
           System.out.println("Failed to load target images: " + targetImage);
           System.exit(1);
       }

       Image source=new Image()
            .withBytes(sourceImageBytes);
       Image target=new Image()
            .withBytes(targetImageBytes);

       CompareFacesRequest request = new CompareFacesRequest()
               .withSourceImage(source)
```

```
                .withTargetImage(target)
                .withSimilarityThreshold(similarityThreshold);

        // Call operation
        CompareFacesResult
 compareFacesResult=rekognitionClient.compareFaces(request);


        // Display results
        List <CompareFacesMatch> faceDetails =
 compareFacesResult.getFaceMatches();
        for (CompareFacesMatch match: faceDetails){
          ComparedFace face= match.getFace();
          BoundingBox position = face.getBoundingBox();
          System.out.println("Face at " + position.getLeft().toString()
                + " " + position.getTop()
                + " matches with " + match.getSimilarity().toString()
                + "% confidence.");

        }
        List<ComparedFace> uncompared = compareFacesResult.getUnmatchedFaces();

        System.out.println("There was " + uncompared.size()
             + " face(s) that did not match");
    }
}
```

Java V2

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

```
import java.util.List;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import software.amazon.awssdk.services.rekognition.model.Image;
import software.amazon.awssdk.services.rekognition.model.BoundingBox;
import software.amazon.awssdk.services.rekognition.model.CompareFacesMatch;
import software.amazon.awssdk.services.rekognition.model.CompareFacesRequest;
import software.amazon.awssdk.services.rekognition.model.CompareFacesResponse;
```

```
import software.amazon.awssdk.services.rekognition.model.ComparedFace;
import software.amazon.awssdk.core.SdkBytes;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;

// snippet-end:[rekognition.java2.detect_faces.import]

/**
 * Before running this Java V2 code example, set up your development
 environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class CompareFaces {

    public static void main(String[] args) {

        final String usage = "\n" +
            "Usage: " +
            "   <pathSource> <pathTarget>\n\n" +
            "Where:\n" +
            "   pathSource - The path to the source image (for example, C:\\AWS\
\pic1.png). \n " +
            "   pathTarget - The path to the target image (for example, C:\\AWS\
\pic2.png). \n\n";

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        Float similarityThreshold = 70F;
        String sourceImage = args[0];
        String targetImage = args[1];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
            .region(region)
            .credentialsProvider(ProfileCredentialsProvider.create("profile-
name"))
            .build();
```

```
        compareTwoFaces(rekClient, similarityThreshold, sourceImage,
targetImage);
        rekClient.close();
    }

    // snippet-start:[rekognition.java2.compare_faces.main]
    public static void compareTwoFaces(RekognitionClient rekClient, Float
similarityThreshold, String sourceImage, String targetImage) {
        try {
            InputStream sourceStream = new FileInputStream(sourceImage);
            InputStream tarStream = new FileInputStream(targetImage);
            SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);
            SdkBytes targetBytes = SdkBytes.fromInputStream(tarStream);

            // Create an Image object for the source image.
            Image souImage = Image.builder()
                .bytes(sourceBytes)
                .build();

            Image tarImage = Image.builder()
                .bytes(targetBytes)
                .build();

            CompareFacesRequest facesRequest = CompareFacesRequest.builder()
                .sourceImage(souImage)
                .targetImage(tarImage)
                .similarityThreshold(similarityThreshold)
                .build();

            // Compare the two images.
            CompareFacesResponse compareFacesResult =
rekClient.compareFaces(facesRequest);
            List<CompareFacesMatch> faceDetails =
compareFacesResult.faceMatches();
            for (CompareFacesMatch match: faceDetails){
                ComparedFace face= match.face();
                BoundingBox position = face.boundingBox();
                System.out.println("Face at " + position.left().toString()
                        + " " + position.top()
                        + " matches with " + face.confidence().toString()
                        + "% confidence.");

            }
```

```
            List<ComparedFace> uncompared = compareFacesResult.unmatchedFaces();
            System.out.println("There was " + uncompared.size() + " face(s) that
 did not match");
            System.out.println("Source image rotation: " +
 compareFacesResult.sourceImageOrientationCorrection());
            System.out.println("target image rotation: " +
 compareFacesResult.targetImageOrientationCorrection());

        } catch(RekognitionException | FileNotFoundException e) {
            System.out.println("Failed to load source image " + sourceImage);
            System.exit(1);
        }
    }
    // snippet-end:[rekognition.java2.compare_faces.main]
}
```

AWS CLI

En este ejemplo, se muestra el resultado JSON de la `compare-faces` AWS CLI operación.

Sustituya `amzn-s3-demo-bucket` por el nombre del bucket de Amazon S3 que contiene las imágenes de origen y destino. Reemplace `source.jpg` y `target.jpg` por los nombres de archivo de las imágenes de origen y destino.

```
aws rekognition compare-faces --target-image \
"{"S3Object":{"Bucket":"amzn-s3-demo-bucket","Name":"image-name"}}" \
--source-image "{"S3Object":{"Bucket":"amzn-s3-demo-bucket","Name":"image-
name"}}"
--profile profile-name
```

Si accede a la CLI en un dispositivo Windows, utilice comillas dobles en lugar de comillas simples y evite las comillas dobles internas con una barra invertida (es decir, \) para corregir cualquier error del analizador que pueda encontrar. Por ver un ejemplo, consulte lo siguiente:

```
aws rekognition compare-faces --target-image "{\"S3Object\":{\"Bucket\":\"amzn-
s3-demo-bucket\",\"Name\":\"image-name\"}}" \
--source-image "{\"S3Object\":{\"Bucket\":\"amzn-s3-demo-bucket\",\"Name\":
\"image-name\"}}" --profile profile-name
```

Python

Este ejemplo muestra información sobre rostros coincidentes de las imágenes de origen y de destino que se cargan desde el sistema de archivos local.

Reemplace los valores de `source_file` y `target_file` por la ruta y el nombre de archivo de las imágenes de origen y de destino. Sustituya el valor de `profile_name` en la línea que crea la sesión de Rekognition por el nombre de su perfil de desarrollador.

```
# Copyright 2018 Amazon.com, Inc. or its affiliates. All Rights Reserved.
# PDX-License-Identifier: MIT-0 (For details, see https://github.com/awsdocs/
amazon-rekognition-developer-guide/blob/master/LICENSE-SAMPLECODE.)

import boto3

def compare_faces(sourceFile, targetFile):

    session = boto3.Session(profile_name='profile-name')
    client = session.client('rekognition')

    imageSource = open(sourceFile, 'rb')
    imageTarget = open(targetFile, 'rb')

    response = client.compare_faces(SimilarityThreshold=80,
                                    SourceImage={'Bytes': imageSource.read()},
                                    TargetImage={'Bytes': imageTarget.read()})

    for faceMatch in response['FaceMatches']:
        position = faceMatch['Face']['BoundingBox']
        similarity = str(faceMatch['Similarity'])
        print('The face at ' +
              str(position['Left']) + ' ' +
              str(position['Top']) +
              ' matches with ' + similarity + '% confidence')

    imageSource.close()
    imageTarget.close()
    return len(response['FaceMatches'])

def main():
    source_file = 'source-file-name'
    target_file = 'target-file-name'
```

```
    face_matches = compare_faces(source_file, target_file)
    print("Face matches: " + str(face_matches))

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

.NET

Este ejemplo muestra información sobre rostros coincidentes de las imágenes de origen y de destino que se cargan desde el sistema de archivos local.

Reemplace los valores de `sourceImage` y `targetImage` por la ruta y el nombre de archivo de las imágenes de origen y de destino.

```
//Copyright 2018 Amazon.com, Inc. or its affiliates. All Rights Reserved.
//PDX-License-Identifier: MIT-0 (For details, see https://github.com/awsdocs/
amazon-rekognition-developer-guide/blob/master/LICENSE-SAMPLECODE.)

using System;
using System.IO;
using Amazon.Rekognition;
using Amazon.Rekognition.Model;

public class CompareFaces
{
    public static void Example()
    {
        float similarityThreshold = 70F;
        String sourceImage = "source.jpg";
        String targetImage = "target.jpg";

        AmazonRekognitionClient rekognitionClient = new
 AmazonRekognitionClient();

        Amazon.Rekognition.Model.Image imageSource = new
 Amazon.Rekognition.Model.Image();
        try
        {
            using (FileStream fs = new FileStream(sourceImage, FileMode.Open,
 FileAccess.Read))
            {
                byte[] data = new byte[fs.Length];
                fs.Read(data, 0, (int)fs.Length);
```

```
                imageSource.Bytes = new MemoryStream(data);
            }
        }
        catch (Exception)
        {
            Console.WriteLine("Failed to load source image: " + sourceImage);
            return;
        }

        Amazon.Rekognition.Model.Image imageTarget = new
Amazon.Rekognition.Model.Image();
        try
        {
            using (FileStream fs = new FileStream(targetImage, FileMode.Open,
FileAccess.Read))
            {
                byte[] data = new byte[fs.Length];
                data = new byte[fs.Length];
                fs.Read(data, 0, (int)fs.Length);
                imageTarget.Bytes = new MemoryStream(data);
            }
        }
        catch (Exception)
        {
            Console.WriteLine("Failed to load target image: " + targetImage);
            return;
        }

        CompareFacesRequest compareFacesRequest = new CompareFacesRequest()
        {
            SourceImage = imageSource,
            TargetImage = imageTarget,
            SimilarityThreshold = similarityThreshold
        };

        // Call operation
        CompareFacesResponse compareFacesResponse =
rekognitionClient.CompareFaces(compareFacesRequest);

        // Display results
        foreach(CompareFacesMatch match in compareFacesResponse.FaceMatches)
        {
            ComparedFace face = match.Face;
            BoundingBox position = face.BoundingBox;
```

```
            Console.WriteLine("Face at " + position.Left
                  + " " + position.Top
                  + " matches with " + match.Similarity
                  + "% confidence.");
        }

        Console.WriteLine("There was " +
 compareFacesResponse.UnmatchedFaces.Count + " face(s) that did not match");

    }
}
```

Ruby

Este ejemplo muestra información sobre rostros coincidentes de las imágenes de origen y de destino que se cargan desde el sistema de archivos local.

Reemplace los valores de `photo_source` y `photo_target` por la ruta y el nombre de archivo de las imágenes de origen y de destino.

```
 # Add to your Gemfile
  # gem 'aws-sdk-rekognition'
  require 'aws-sdk-rekognition'
  credentials = Aws::Credentials.new(
     ENV['AWS_ACCESS_KEY_ID'],
     ENV['AWS_SECRET_ACCESS_KEY']
  )
  bucket        = 'bucket' # the bucketname without s3://
  photo_source  = 'source.jpg'
  photo_target  = 'target.jpg'
  client   = Aws::Rekognition::Client.new credentials: credentials
  attrs = {
    source_image: {
      s3_object: {
        bucket: bucket,
        name: photo_source
      },
    },
    target_image: {
      s3_object: {
        bucket: bucket,
        name: photo_target
      },
```

```
    },
    similarity_threshold: 70
  }
  response = client.compare_faces attrs
  response.face_matches.each do |face_match|
    position   = face_match.face.bounding_box
    similarity = face_match.similarity
    puts "The face at: #{position.left}, #{position.top} matches with
 #{similarity} % confidence"
  end
```

Node.js

Este ejemplo muestra información sobre rostros coincidentes de las imágenes de origen y de destino que se cargan desde el sistema de archivos local.

Reemplace los valores de `photo_source` y `photo_target` por la ruta y el nombre de archivo de las imágenes de origen y de destino. Sustituya el valor de `profile_name` en la línea que crea la sesión de Rekognition por el nombre de su perfil de desarrollador.

```
// Load the SDK
var AWS = require('aws-sdk');
const bucket = 'bucket-name' // the bucket name without s3://
const photo_source  = 'photo-source-name' // path and the name of file
const photo_target = 'photo-target-name'

var credentials = new AWS.SharedIniFileCredentials({profile: 'profile-name'});
AWS.config.credentials = credentials;
AWS.config.update({region:'region-name'});

const client = new AWS.Rekognition();
   const params = {
     SourceImage: {
       S3Object: {
         Bucket: bucket,
         Name: photo_source
       },
     },
     TargetImage: {
       S3Object: {
         Bucket: bucket,
         Name: photo_target
       },
```

```
    },
    SimilarityThreshold: 70
  }
  client.compareFaces(params, function(err, response) {
    if (err) {
      console.log(err, err.stack); // an error occurred
    } else {
      response.FaceMatches.forEach(data => {
        let position   = data.Face.BoundingBox
        let similarity = data.Similarity
        console.log(`The face at: ${position.Left}, ${position.Top} matches
with ${similarity} % confidence`)
      }) // for response.faceDetails
    } // if
  });
```

CompareFaces solicitud de operación

La entrada de `CompareFaces` es una imagen. En este ejemplo, las imágenes de origen y destino se cargan desde el sistema de archivos local. El parámetro `SimilarityThreshold` de entrada especifica el mínimo de confianza que la comparación de rostros debe tener para incluirse en la respuesta. Para obtener más información, consulte Trabajar con imágenes.

```
{
    "SourceImage": {
        "Bytes": "/9j/4AAQSk2Q==..."
    },
    "TargetImage": {
        "Bytes": "/9j/4O1Q==..."
    },
    "SimilarityThreshold": 70
}
```

CompareFaces respuesta de operación

La respuesta incluye:

- Una matriz de los rostros coincidentes: una lista de los rostros coincidentes con puntuaciones de similitud y metadatos para cada uno de ellos. Si coinciden varios rostros, la matriz `faceMatches` incluye todos los rostros coincidentes.

- Detalles de la coincidencia de rostros: con cada rostro coincidente se proporciona también un cuadro delimitador, un valor de confianza, ubicaciones de referencia y una puntuación de similitud.
- Lista de rostros sin coincidencias: la respuesta también incluye los rostros de la imagen de destino que no coinciden con el rostro de la imagen de origen. Incluye un cuadro delimitador para cada rostro sin coincidencia.
- Información del rostro de origen: incluye información sobre el rostro de la imagen de origen que se ha utilizado para la comparación, incluido el cuadro delimitador y el valor de confianza.

El ejemplo muestra que se ha encontrado una coincidencia de rostro en la imagen de destino. Para esa coincidencia de rostro, proporciona un cuadro delimitador y un valor de confianza (el nivel de confianza que tiene Amazon Rekognition de que el cuadro delimitador contenga un rostro). La puntuación de similitud de 99,99 indica la similitud entre los rostros. El ejemplo también muestra un rostro que Amazon Rekognition ha encontrado en la imagen de destino y que no coincide con el rostro analizado en la imagen de origen.


```
{
    "FaceMatches": [{
        "Face": {
            "BoundingBox": {
                "Width": 0.5521978139877319,
                "Top": 0.1203877404332161,
                "Left": 0.23626373708248138,
                "Height": 0.3126954436302185
            },
            "Confidence": 99.98751068115234,
            "Pose": {
                "Yaw": -82.36799621582031,
                "Roll": -62.13221740722656,
                "Pitch": 0.8652129173278809
            },
            "Quality": {
                "Sharpness": 99.99880981445312,
                "Brightness": 54.49755096435547
            },
            "Landmarks": [{
                    "Y": 0.2996366024017334,
                    "X": 0.41685718297958374,
                    "Type": "eyeLeft"
                },
                {
```

```
                    "Y": 0.2658946216106415,
                    "X": 0.4414493441581726,
                    "Type": "eyeRight"
                },
                {
                    "Y": 0.3465650677680969,
                    "X": 0.48636093735694885,
                    "Type": "nose"
                },
                {
                    "Y": 0.30935320258140564,
                    "X": 0.6251809000968933,
                    "Type": "mouthLeft"
                },
                {
                    "Y": 0.26942989230155945,
                    "X": 0.6454493403434753,
                    "Type": "mouthRight"
                }
            ]
        },
        "Similarity": 100.0
    }],
    "SourceImageOrientationCorrection": "ROTATE_90",
    "TargetImageOrientationCorrection": "ROTATE_90",
    "UnmatchedFaces": [{
        "BoundingBox": {
            "Width": 0.4890109896659851,
            "Top": 0.6566604375839233,
            "Left": 0.10989011079072952,
            "Height": 0.278298944234848
        },
        "Confidence": 99.99992370605469,
        "Pose": {
            "Yaw": 51.51519012451172,
            "Roll": -110.32493591308594,
            "Pitch": -2.322134017944336
        },
        "Quality": {
            "Sharpness": 99.99671173095703,
            "Brightness": 57.23163986206055
        },
        "Landmarks": [{
                "Y": 0.8288310766220093,
```

```
                "X": 0.3133862614631653,
                "Type": "eyeLeft"
            },
            {
                "Y": 0.7632885575294495,
                "X": 0.28091415762901306,
                "Type": "eyeRight"
            },
            {
                "Y": 0.7417283654212952,
                "X": 0.3631140887737274,
                "Type": "nose"
            },
            {
                "Y": 0.8081989884376526,
                "X": 0.4856561422348022 5,
                "Type": "mouthLeft"
            },
            {
                "Y": 0.7548204660415649,
                "X": 0.46090251207351685,
                "Type": "mouthRight"
            }
        ]
    }],
    "SourceImageFace": {
        "BoundingBox": {
            "Width": 0.5521978139877319,
            "Top": 0.1203877404332161,
            "Left": 0.23626373708248138,
            "Height": 0.3126954436302185
        },
        "Confidence": 99.98751068115234
    }
}
```


Detección de rostros en un vídeo almacenado

Amazon Rekognition Video puede detectar rostros en vídeos almacenados en un bucket de Amazon S3 y proporcionar información como:

- Las veces que los rostros se detectan en un vídeo.

- La ubicación de los rostros en un fotograma de vídeo a la hora en la que se detectan.
- Referencias faciales como, por ejemplo, la posición del ojo izquierdo.
- Atributos adicionales, tal y como se explica en la página the section called “Directrices sobre los atributos faciales”.

La detección de rostros de Amazon Rekognition Video en vídeos almacenados una operación asíncrona. Para iniciar la detección de rostros en los videos, llame StartFaceDetection. Amazon Rekognition Video publica el estado de finalización de una operación de análisis de vídeo en un tema de Amazon Simple Notification Service (Amazon SNS). Si el análisis de vídeo es correcto, puede llamar a GetFaceDetection para obtener los resultados del análisis de vídeo. Para obtener más información sobre cómo iniciar el análisis de vídeo y obtener los resultados, consulte Cómo llamar a las operaciones de Amazon Rekognition Video.

Este procedimiento amplía el código de Análisis de un vídeo almacenado en un bucket de Amazon S3 con Java o Python (SDK), que utiliza una cola de Amazon Simple Queue Service (Amazon SQS) para obtener el estado de realización de una solicitud de análisis de vídeo.

Para detectar rostros en un vídeo almacenado en un bucket de Amazon S3 (SDK)

1. Realice Análisis de un vídeo almacenado en un bucket de Amazon S3 con Java o Python (SDK).
2. Añada el código siguiente a la clase `VideoDetect` que ha creado en el paso 1.

   AWS CLI

   - En el siguiente ejemplo de código, cambie `amzn-s3-demo-bucket` y `video-name` por el nombre del bucket de Amazon S3 y el nombre de archivo que especificó en el paso 2.
   - Cambie `region-name` por la región de AWS que está utilizando. Sustituya el valor de `profile_name` de por el nombre de su perfil de desarrollador.
   - Reemplace `TopicARN` por el ARN del tema de Amazon SNS que creó en el paso 3 de Configuración de Amazon Rekognition Video.
   - Cambie `RoleARN` por el ARN del rol de servicio de IAM que creó en el paso 7 de Configuración de Amazon Rekognition Video.

```
aws rekognition start-face-detection --video "{"S3Object":{"Bucket":"amzn-s3-
demo-bucket","Name":"Video-Name"}}" --notification-channel
```

```
"{"SNSTopicArn":"Topic-ARN","RoleArn":"Role-ARN"}" --region region-name --
profile profile-name
```

Si accede a la CLI en un dispositivo Windows, utilice comillas dobles en lugar de comillas simples y evite las comillas dobles internas con una barra invertida (es decir, \) para corregir cualquier error del analizador que pueda encontrar. Por ver un ejemplo, consulte lo siguiente:

```
aws rekognition start-face-detection --video "{\"S3Object\":{\"Bucket\":\"amzn-
s3-demo-bucket\",\"Name\":\"Video-Name\"}}" --notification-channel
"{\"SNSTopicArn\":\"Topic-ARN\",\"RoleArn\":\"Role-ARN\"}" --region region-name
 --profile profile-name
```

Tras ejecutar la operación `StartFaceDetection` y obtener el número de ID del trabajo, ejecute la siguiente operación `GetFaceDetection` e indique el número de ID del trabajo:

```
aws rekognition get-face-detection --job-id job-id-number  --profile profile-
name
```

Java

```
//Copyright 2018 Amazon.com, Inc. or its affiliates. All Rights Reserved.
//PDX-License-Identifier: MIT-0 (For details, see https://github.com/awsdocs/
amazon-rekognition-developer-guide/blob/master/LICENSE-SAMPLECODE.)


private static void StartFaceDetection(String bucket, String video) throws
 Exception{

    NotificationChannel channel= new NotificationChannel()
            .withSNSTopicArn(snsTopicArn)
            .withRoleArn(roleArn);

    StartFaceDetectionRequest req = new StartFaceDetectionRequest()
            .withVideo(new Video()
                    .withS3Object(new S3Object()
                        .withBucket(bucket)
```

```
                        .withName(video)))
            .withNotificationChannel(channel);



    StartFaceDetectionResult startLabelDetectionResult =
 rek.startFaceDetection(req);
    startJobId=startLabelDetectionResult.getJobId();

}

private static void GetFaceDetectionResults() throws Exception{

    int maxResults=10;
    String paginationToken=null;
    GetFaceDetectionResult faceDetectionResult=null;

    do{
        if (faceDetectionResult !=null){
            paginationToken = faceDetectionResult.getNextToken();
        }

        faceDetectionResult = rek.getFaceDetection(new
 GetFaceDetectionRequest()
             .withJobId(startJobId)
             .withNextToken(paginationToken)
             .withMaxResults(maxResults));

        VideoMetadata videoMetaData=faceDetectionResult.getVideoMetadata();

        System.out.println("Format: " + videoMetaData.getFormat());
        System.out.println("Codec: " + videoMetaData.getCodec());
        System.out.println("Duration: " + videoMetaData.getDurationMillis());
        System.out.println("FrameRate: " + videoMetaData.getFrameRate());


        //Show faces, confidence and detection times
        List<FaceDetection> faces= faceDetectionResult.getFaces();

        for (FaceDetection face: faces) {
            long seconds=face.getTimestamp()/1000;
            System.out.print("Sec: " + Long.toString(seconds) + " ");
            System.out.println(face.getFace().toString());
            System.out.println();
```

```
        }
    } while (faceDetectionResult !=null && faceDetectionResult.getNextToken() !=
 null);


}
```

En la función `main`, reemplace las líneas:

```
        StartLabelDetection(amzn-s3-demo-bucket, video);

        if (GetSQSMessageSuccess()==true)
         GetLabelDetectionResults();
```

por:

```
        StartFaceDetection(amzn-s3-demo-bucket, video);

        if (GetSQSMessageSuccess()==true)
         GetFaceDetectionResults();
```

Java V2

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

```
//snippet-start:[rekognition.java2.recognize_video_faces.import]
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.*;
import java.util.List;
//snippet-end:[rekognition.java2.recognize_video_faces.import]


/**
* Before running this Java V2 code example, set up your development environment,
 including your credentials.
*
* For more information, see the following documentation topic:
*
```

```
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
*/
public class VideoDetectFaces {

 private static String startJobId ="";
 public static void main(String[] args) {

     final String usage = "\n" +
         "Usage: " +
         "   <bucket> <video> <topicArn> <roleArn>\n\n" +
         "Where:\n" +
         "   bucket - The name of the bucket in which the video is located (for
 example, (for example, amzn-s3-demo-bucket). \n\n"+
         "   video - The name of video (for example, people.mp4). \n\n" +
         "   topicArn - The ARN of the Amazon Simple Notification Service
 (Amazon SNS) topic. \n\n" +
         "   roleArn - The ARN of the AWS Identity and Access Management (IAM)
 role to use. \n\n" ;

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

     String bucket = args[0];
     String video = args[1];
     String topicArn = args[2];
     String roleArn = args[3];

     Region region = Region.US_EAST_1;
     RekognitionClient rekClient = RekognitionClient.builder()
         .region(region)
         .credentialsProvider(ProfileCredentialsProvider.create("profile-name"))
         .build();

     NotificationChannel channel = NotificationChannel.builder()
         .snsTopicArn(topicArn)
         .roleArn(roleArn)
         .build();

     StartFaceDetection(rekClient, channel, bucket, video);
     GetFaceResults(rekClient);
     System.out.println("This example is done!");
```

```
    rekClient.close();
}

// snippet-start:[rekognition.java2.recognize_video_faces.main]
public static void StartFaceDetection(RekognitionClient rekClient,
                                       NotificationChannel channel,
                                       String bucket,
                                       String video) {

    try {
        S3Object s3Obj = S3Object.builder()
            .bucket(bucket)
            .name(video)
            .build();

        Video vidOb = Video.builder()
            .s3Object(s3Obj)
            .build();

        StartFaceDetectionRequest  faceDetectionRequest =
StartFaceDetectionRequest.builder()
            .jobTag("Faces")
            .faceAttributes(FaceAttributes.ALL)
            .notificationChannel(channel)
            .video(vidOb)
            .build();

        StartFaceDetectionResponse startLabelDetectionResult =
rekClient.startFaceDetection(faceDetectionRequest);
        startJobId=startLabelDetectionResult.jobId();

    } catch(RekognitionException e) {
        System.out.println(e.getMessage());
        System.exit(1);
    }
}

public static void GetFaceResults(RekognitionClient rekClient) {

    try {
        String paginationToken=null;
        GetFaceDetectionResponse faceDetectionResponse=null;
        boolean finished = false;
        String status;
```

```
        int yy=0 ;

        do{
            if (faceDetectionResponse !=null)
                paginationToken = faceDetectionResponse.nextToken();

            GetFaceDetectionRequest recognitionRequest =
GetFaceDetectionRequest.builder()
                .jobId(startJobId)
                .nextToken(paginationToken)
                .maxResults(10)
                .build();

            // Wait until the job succeeds
            while (!finished) {

                faceDetectionResponse =
rekClient.getFaceDetection(recognitionRequest);
                status = faceDetectionResponse.jobStatusAsString();

                if (status.compareTo("SUCCEEDED") == 0)
                    finished = true;
                else {
                    System.out.println(yy + " status is: " + status);
                    Thread.sleep(1000);
                }
                yy++;
            }

            finished = false;

            // Proceed when the job is done - otherwise VideoMetadata is null
            VideoMetadata videoMetaData=faceDetectionResponse.videoMetadata();
            System.out.println("Format: " + videoMetaData.format());
            System.out.println("Codec: " + videoMetaData.codec());
            System.out.println("Duration: " + videoMetaData.durationMillis());
            System.out.println("FrameRate: " + videoMetaData.frameRate());
            System.out.println("Job");

            // Show face information
            List<FaceDetection> faces= faceDetectionResponse.faces();

            for (FaceDetection face: faces) {
                String age = face.face().ageRange().toString();
```

```
                String smile = face.face().smile().toString();
                System.out.println("The detected face is estimated to be"
                            + age + " years old.");
                System.out.println("There is a smile : "+smile);
             }

         } while (faceDetectionResponse !=null &&
 faceDetectionResponse.nextToken() != null);

     } catch(RekognitionException | InterruptedException e) {
         System.out.println(e.getMessage());
         System.exit(1);
     }
 }
 // snippet-end:[rekognition.java2.recognize_video_faces.main]
}
```

Python

```
#Copyright 2018 Amazon.com, Inc. or its affiliates. All Rights Reserved.
#PDX-License-Identifier: MIT-0 (For details, see https://github.com/awsdocs/
amazon-rekognition-developer-guide/blob/master/LICENSE-SAMPLECODE.)

    # ============== Faces===============
    def StartFaceDetection(self):
        response=self.rek.start_face_detection(Video={'S3Object': {'Bucket':
 self.bucket, 'Name': self.video}},
            NotificationChannel={'RoleArn': self.roleArn, 'SNSTopicArn':
 self.snsTopicArn})

        self.startJobId=response['JobId']
        print('Start Job Id: ' + self.startJobId)

    def GetFaceDetectionResults(self):
        maxResults = 10
        paginationToken = ''
        finished = False

        while finished == False:
            response = self.rek.get_face_detection(JobId=self.startJobId,
                                            MaxResults=maxResults,
                                            NextToken=paginationToken)
```

```
            print('Codec: ' + response['VideoMetadata']['Codec'])
            print('Duration: ' + str(response['VideoMetadata']
['DurationMillis']))
            print('Format: ' + response['VideoMetadata']['Format'])
            print('Frame rate: ' + str(response['VideoMetadata']['FrameRate']))
            print()

            for faceDetection in response['Faces']:
                print('Face: ' + str(faceDetection['Face']))
                print('Confidence: ' + str(faceDetection['Face']['Confidence']))
                print('Timestamp: ' + str(faceDetection['Timestamp']))
                print()

            if 'NextToken' in response:
                paginationToken = response['NextToken']
            else:
                finished = True
```

En la función `main`, reemplace las líneas:

```
    analyzer.StartLabelDetection()
    if analyzer.GetSQSMessageSuccess()==True:
        analyzer.GetLabelDetectionResults()
```

por:

```
    analyzer.StartFaceDetection()
    if analyzer.GetSQSMessageSuccess()==True:
        analyzer.GetFaceDetectionResults()
```

> **Note**
>
> Si ya ha ejecutado un ejemplo de vídeo distinto de Análisis de un vídeo almacenado en un bucket de Amazon S3 con Java o Python (SDK), el nombre de la función que se va a reemplazar es distinto.

3. Ejecute el código. Se muestra información sobre los rostros detectados en el vídeo.

GetFaceDetection respuesta de operación

`GetFaceDetection` devuelve una matriz (`Faces`) que contiene información sobre los rostros detectados en el vídeo. Un elemento de matriz, FaceDetection, existe para cada vez que se detecta un rostro en el vídeo. Los elementos de la matriz se devuelven ordenados por tiempo, en milisegundos desde el inicio del vídeo.

El siguiente ejemplo es una respuesta JSON parcial desde `GetFaceDetection`. En la respuesta, tenga en cuenta lo siguiente:

- Cuadro delimitador: las coordenadas del cuadro delimitador que rodea el rostro.
- Confianza: grado de confianza de que el cuadro delimitador contiene un rostro.
- Referencias faciales: una matriz de referencias faciales. Para cada referencia (como el ojo izquierdo, el ojo derecho y la boca), la respuesta proporciona las coordenadas `x` e `y`.
- Atributos faciales: conjunto de atributos faciales, que incluye: barba AgeRange, emociones, anteojos, género, EyesOpen bigote MouthOpen, sonrisa y gafas de sol. El valor puede ser de diferentes tipos, como un tipo booleano (si una persona lleva gafas), una cadena (si la persona es un hombre o una mujer), etc. Además, para la mayoría de los atributos la respuesta proporciona también una confianza en el valor detectado para el atributo. Ten en cuenta que, si bien EyeDirection los atributos FaceOccluded y los atributos son compatibles cuando se utilizan`DetectFaces`, no se admiten cuando se analizan vídeos con y. `StartFaceDetection` `GetFaceDetection`
- Marca de tiempo: hora a la que se detectó el rostro en el vídeo.
- Información de paginación: el ejemplo muestra una página de información de detección de rostros. Puede especificar la cantidad de elementos de persona que se van a devolver en el parámetro de entrada `MaxResults` para `GetFaceDetection`. Si existen más resultados que `MaxResults`, `GetFaceDetection` devuelve un token (`NextToken`) que se utiliza para obtener la siguiente página de resultados. Para obtener más información, consulte Obtención de los resultados del análisis de Amazon Rekognition Video.
- Información de vídeo: la respuesta incluye información acerca del formato de vídeo (`VideoMetadata`) de cada página de información devuelta por `GetFaceDetection`.
- Calidad: describe el brillo y la nitidez del rostro.
- Postura: describe la rotación del rostro.

```
{
```

```
    "Faces": [
        {
            "Face": {
                "BoundingBox": {
                    "Height": 0.23000000417232513,
                    "Left": 0.42500001192092896,
                    "Top": 0.16333332657814026,
                    "Width": 0.12937499582767487
                },
                "Confidence": 99.97504425048828,
                "Landmarks": [
                    {
                        "Type": "eyeLeft",
                        "X": 0.46415066719055176,
                        "Y": 0.2572723925113678
                    },
                    {
                        "Type": "eyeRight",
                        "X": 0.5068183541297913,
                        "Y": 0.23705792427062988
                    },
                    {
                        "Type": "nose",
                        "X": 0.49765899777412415,
                        "Y": 0.28383663296699524
                    },
                    {
                        "Type": "mouthLeft",
                        "X": 0.487221896648407,
                        "Y": 0.3452930748462677
                    },
                    {
                        "Type": "mouthRight",
                        "X": 0.5142884850502014,
                        "Y": 0.33167609572410583
                    }
                ],
                "Pose": {
                    "Pitch": 15.966927528381348,
                    "Roll": -15.547388076782227,
                    "Yaw": 11.34195613861084
                },
                "Quality": {
                    "Brightness": 44.80223083496094,
```

```
                    "Sharpness": 99.95819854736328
                }
            },
            "Timestamp": 0
        },
        {
            "Face": {
                "BoundingBox": {
                    "Height": 0.20000000298023224,
                    "Left": 0.029999999329447746,
                    "Top": 0.2199999988079071,
                    "Width": 0.11249999701976776
                },
                "Confidence": 99.85971069335938,
                "Landmarks": [
                    {
                        "Type": "eyeLeft",
                        "X": 0.06842322647571564,
                        "Y": 0.3010137975215912
                    },
                    {
                        "Type": "eyeRight",
                        "X": 0.10543643683195114,
                        "Y": 0.29697132110595703
                    },
                    {
                        "Type": "nose",
                        "X": 0.09569807350635529,
                        "Y": 0.33701086044311523
                    },
                    {
                        "Type": "mouthLeft",
                        "X": 0.0732642263174057,
                        "Y": 0.3757539987564087
                    },
                    {
                        "Type": "mouthRight",
                        "X": 0.10589495301246643,
                        "Y": 0.3722417950630188
                    }
                ],
                "Pose": {
                    "Pitch": -0.5589138865470886,
                    "Roll": -5.1093974113464355,
```

```
                    "Yaw": 18.69594955444336
                },
                "Quality": {
                    "Brightness": 43.052337646484375,
                    "Sharpness": 99.68138885498047
                }
            },
            "Timestamp": 0
        },
        {
            "Face": {
                "BoundingBox": {
                    "Height": 0.2177777737379074,
                    "Left": 0.7593749761581421,
                    "Top": 0.13333334028720856,
                    "Width": 0.12250000238418579
                },
                "Confidence": 99.63436889648438,
                "Landmarks": [
                    {
                        "Type": "eyeLeft",
                        "X": 0.8005779385566711,
                        "Y": 0.20915353298187256
                    },
                    {
                        "Type": "eyeRight",
                        "X": 0.8391435146331787,
                        "Y": 0.21049551665782928
                    },
                    {
                        "Type": "nose",
                        "X": 0.8191410899162292,
                        "Y": 0.2523227035999298
                    },
                    {
                        "Type": "mouthLeft",
                        "X": 0.8093273043632507,
                        "Y": 0.29053622484207153
                    },
                    {
                        "Type": "mouthRight",
                        "X": 0.8366993069648743,
                        "Y": 0.29101791977882385
                    }
```

```
                ],
                "Pose": {
                    "Pitch": 3.165884017944336,
                    "Roll": 1.4182015657424927,
                    "Yaw": -11.151537895202637
                },
                "Quality": {
                    "Brightness": 28.910892486572266,
                    "Sharpness": 97.61507415771484
                }
            },
            "Timestamp": 0
        }.......

    ],
    "JobStatus": "SUCCEEDED",
    "NextToken": "i7fj5XPV/
fwviXqz0eag9Ow332Jd5G8ZGWf7hooirD/6V1qFmjKFOQZ6QPWUiqv29HbyuhMNqQ==",
    "VideoMetadata": {
        "Codec": "h264",
        "DurationMillis": 67301,
        "FileExtension": "mp4",
        "Format": "QuickTime / MOV",
        "FrameHeight": 1080,
        "FrameRate": 29.970029830932617,
        "FrameWidth": 1920
    }
}
```

Búsqueda de rostros en una colección

Amazon Rekognition le permite usar un rostro de entrada para buscar coincidencias en una colección de rostros almacenados. Empiece por almacenar la información sobre los rostros detectados en contenedores del lado del servidor denominados «colecciones». Las colecciones almacenan tanto caras como usuarios individuales (algunas caras de la misma persona). Los rostros individuales se almacenan como vectores faciales, una representación matemática del rostro (no una imagen real del rostro). Se pueden usar diferentes imágenes de la misma persona para crear y almacenar varios vectores faciales en la misma colección. Después, puede agregar varios vectores faciales de la misma persona para crear un vector de usuario. Los vectores de usuario pueden ofrecer una mayor precisión en la búsqueda de rostros con representaciones más robustas, que contienen distintos grados de iluminación, nitidez, pose, apariencia, etc.

Una vez que haya creado una colección, puede usar un rostro de entrada para buscar vectores de usuario o vectores de rostros coincidentes en una colección. La búsqueda en vectores de usuarios puede mejorar considerablemente la precisión en comparación con la búsqueda en vectores de rostros individuales. Puede utilizar rostros detectados en imágenes, vídeos almacenados y vídeos en streaming para buscar en los vectores de rostros almacenados. Puede utilizar los rostros detectados en las imágenes para buscar en los vectores de usuario almacenados.

Para almacenar la información del rostro, tiene que hacer lo siguiente:

1. Crear una colección: para almacenar información facial, primero debes crear (CreateCollection) una colección de rostros en una de las AWS regiones de tu cuenta. Esta colección de rostros se especifica cuando se llama a la operación `IndexFaces`.
2. Indexar rostros: la IndexFacesoperación detecta los rostros de una imagen, extrae y almacena los vectores faciales de la colección. Puede usar esta operación para detectar rostros en una imagen y conservar la información sobre los rasgos faciales que se detecten en una colección. Este es un ejemplo de una operación de API con almacenamiento porque el servicio almacena la información de vectores faciales en el servidor.

Para crear un usuario y asociarle varios vectores faciales, debe hacer lo siguiente:

1. Crear un usuario: primero debe crear un usuario con CreateUser. Puede mejorar la precisión de la coincidencia de rostros agregando varios vectores faciales de la misma persona en un vector de usuario. Puede asociar hasta 100 vectores faciales a un vector de usuario.

2. Asociar caras: después de crear el usuario, puede añadir vectores faciales existentes a ese usuario con la AssociateFacesoperación. Los vectores faciales deben residir en la misma colección que un vector de usuario para poder asociarse a ese vector de usuario.

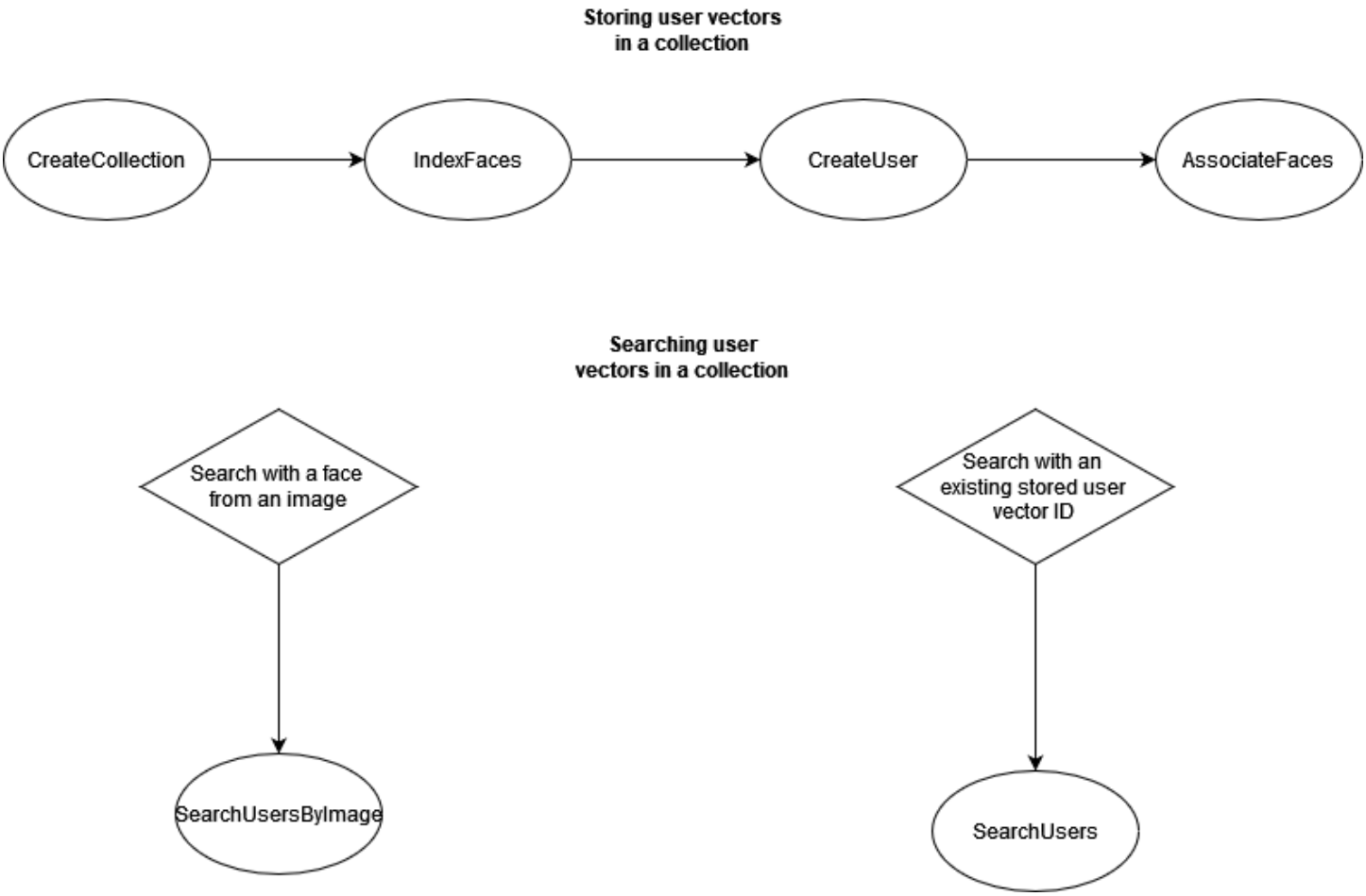
Tras crear una colección y almacenar los vectores faciales y de usuario, puede utilizar las siguientes operaciones para buscar coincidencias faciales:

- SearchFacesByImage- Para buscar rostros individuales almacenados con un rostro de una imagen.
- SearchFaces- Para buscar rostros individuales almacenados con un identificador facial suministrado.
- SearchUsers- Para buscar usuarios almacenados con un identificador facial o un seudónimo proporcionados.
- SearchUsersByImage- Para buscar usuarios almacenados con un rostro de una imagen.
- StartFaceSearch- Para buscar rostros en un vídeo almacenado.
- CreateStreamProcessor- Para buscar rostros en un vídeo en streaming.

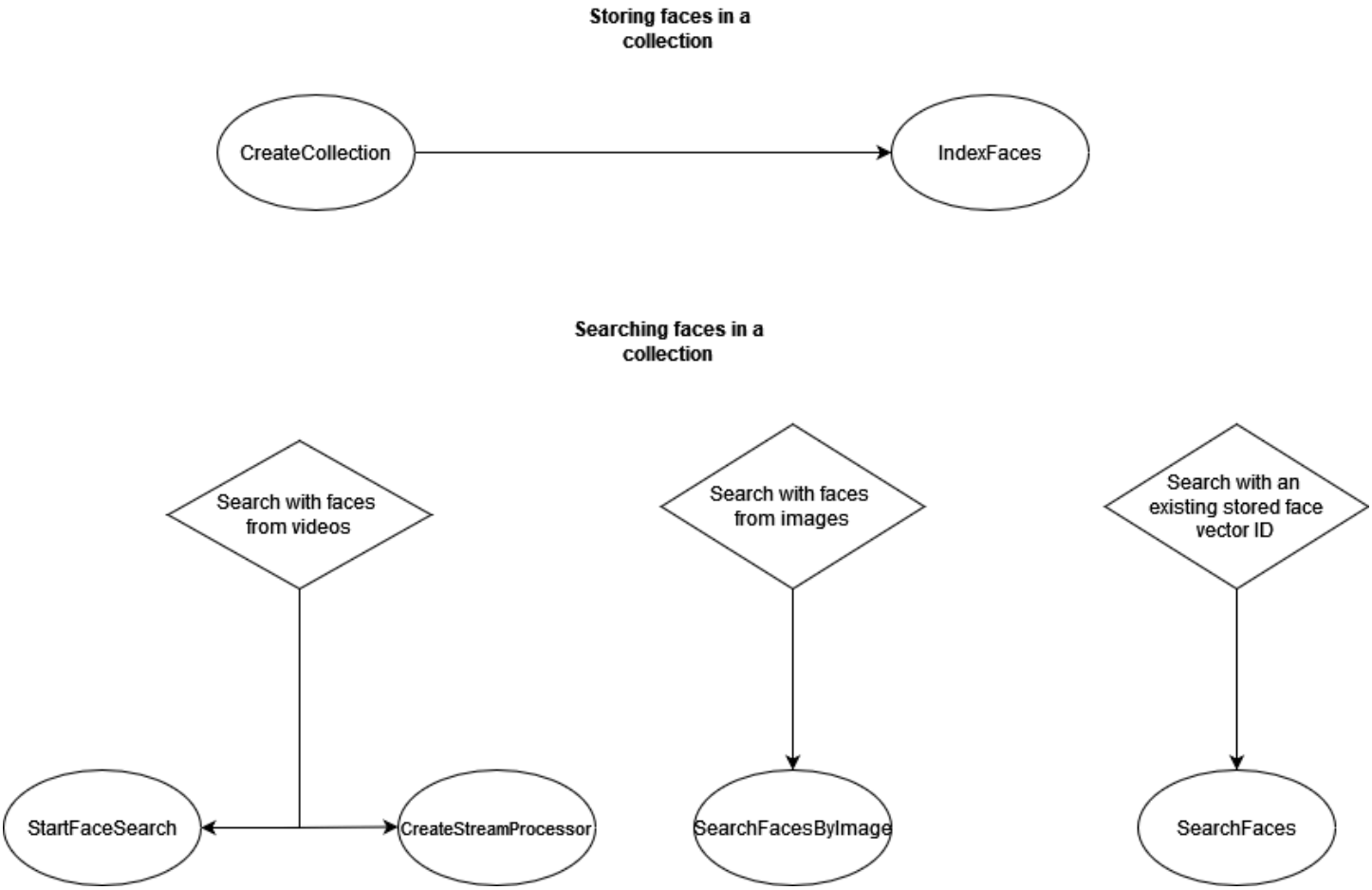
> **Note**
>
> Las colecciones almacenan vectores faciales, que son representaciones matemáticas de los rostros. Las colecciones no almacenan imágenes de rostros.

Los siguientes diagramas muestran el orden de las operaciones de llamada en función de los objetivos de uso de las colecciones por parte del usuario:

Para una máxima precisión de la coincidencia con los vectores de usuario:



Para una coincidencia de alta precisión con vectores faciales individuales:



Puede utilizar las colecciones en diversas situaciones. Por ejemplo, puede crear una colección de rostros que almacene los rostros detectados a partir de las imágenes escaneadas de las tarjetas de identificación de los empleados y emitidas IDs por el gobierno mediante `AssociateFaces` las operaciones `IndexFaces` and. Cuando un empleado entra en el edificio, se captura una imagen del rostro del empleado y se envía a la operación `SearchUsersByImage`. Si el rostro coincidente produce una puntuación de similitud lo suficientemente alta (por ejemplo, un 99%), se puede autenticar al empleado.

Administración de colecciones de rostros, rostros y usuarios

La colección de rostros es el recurso que Amazon Rekognition utiliza para incluir la información relacionada con los rostros y los usuarios. Se utilizan varias operaciones de la API de Rekognition para administrar la colección en sí, además de cualquier rostro o vector facial almacenado en la colección.

Administración de una colección

La colección de rostros es el principal recurso de Amazon Rekognition y cada colección de rostros que se crea tiene un nombre de recurso de Amazon (ARN) único. Creas cada colección de rostros en una AWS región específica de tu cuenta. Cuando se crea una nueva colección, está asociada a la versión más reciente del modelo de detección de rostros. Para obtener más información, consulte Descripción del control de versiones de los modelos.

Puede realizar las siguientes operaciones de administración en una colección:

- Crear una colección con CreateCollection. Para obtener más información, consulte Creación de una colección.
- Enumerar las colecciones disponibles con ListCollections. Para obtener más información, consulte Listado de colecciones.
- Describir una colección con DescribeCollection. Para obtener más información, consulte Descripción de una colección.
- Eliminar una colección con DeleteCollection. Para obtener más información, consulte Eliminación de una colección.

Administración de rostros en una colección

Una vez creada una colección de rostros, puede almacenar rostros en ella. Amazon Rekognition ofrece las siguientes operaciones para la administración de los rostros de una colección:

- La operación IndexFaces detecta rostros en la imagen de entrada (JPEG o PNG) y las añade a la colección de rostros especificada. Se devuelve un ID de rostro único por cada rostro detectado en la imagen. Una vez guardados los rostros, puede buscar rostros coincidentes en la colección. Para obtener más información, consulte Agregar rostros a una colección.
- La operación ListFaces enumera los rostros de una colección. Para obtener más información, consulte Agregar rostros a una colección.
- La operación DeleteFaces elimina los rostros de una colección. Para obtener más información, consulte Eliminación de rostros de una colección.

Administrar usuarios en una colección

Después de almacenar varios vectores de rostros de la misma persona, puede mejorar la precisión asociando todos esos vectores faciales en un vector de usuario. Puede utilizar las siguientes operaciones para administrar sus usuarios:

- CreateUser- La operación crea un nuevo usuario en una colección con un ID de usuario único proporcionado.
- AssociateUsers- Añade de 1 a 100 caras únicas IDs a un seudónimo. Después de asociar al menos un ID de rostro a un usuario, puede buscar coincidencias con ese usuario en su colección.
- ListUsers- Muestra los usuarios de una colección.
- DeleteUsers- Elimina un usuario de una colección con el ID de usuario proporcionado.
- DisassociateFaces- Elimina uno o más rostros IDs de un usuario.

Uso de umbrales de similitud para asociar rostros y buscar coincidencias con estos

Los umbrales de similitud se utilizan tanto para asociar rostros como para buscar coincidencias con estos. A continuación, se incluyen instrucciones para usar los umbrales de similitud en ambos casos de uso.

Uso de umbrales de similitud para asociar rostros

Al asociar rostros mediante esta AssociateFacesoperación, es importante asegurarse de que los rostros que se asocien a un usuario sean todos de la misma persona. Como ayuda, el parámetro `UserMatchThreshold` especifica el nivel mínimo de confianza de coincidencia del usuario necesario para que el nuevo rostro se asocie con un `UserID` que ya contenga al menos un `FaceID`. Esto ayuda a garantizar que `FaceIds` esté asociado con el `UserID` correcto. El valor oscila entre 0 y 100 y el valor predeterminado es 75.

Uso de umbrales de similitud para que coincidan con rostros

Le permitimos controlar los resultados de todas las operaciones de búsqueda (CompareFaces, SearchFaces, SearchFacesByImage, SearchUsers, SearchUsersByImage) proporcionando un umbral de similitud como parámetro de entrada.

`FaceMatchThreshold`, el atributo de entrada del umbral de similitud para `SearchFaces` y `SearchFacesByImage`, controla la cantidad de resultados que se devuelven en función de la similitud con el rostro que se hace coincidir. El atributo de entrada del umbral de similitud para `SearchUsers` y `SearchUsersByImage` es `UserMatchThreshold` controla la cantidad de resultados que se devuelven en función de la similitud con el vector de usuario que se hace coincidir. El atributo de umbral es `SimilarityThreshold` para `CompareFaces`.

Las respuestas con un valor de atributo de respuesta `Similarity` inferior al umbral no se devuelven. Este umbral es importante para calibrar su caso de uso, porque puede determinar la cantidad de falsos positivos que se incluyen en los resultados de coincidencia. Esto controla la exhaustividad de los resultados de la búsqueda; mientras más bajo sea el umbral, mayor será la exhaustividad.

Todos los sistemas de machine learning son probabilísticos. Debe utilizar su buen juicio al establecer el umbral de similitud correcto, en función de su caso de uso. Por ejemplo, si desea crear una aplicación de fotografía para identificar a miembros de la familia parecidos, puede elegir un umbral más bajo (como, por ejemplo, un 80 %). Por otra parte, para muchos casos de uso para cumplimiento de la ley, le recomendamos utilizar un valor de umbral alto del 99 % o superior para reducir las identificaciones erróneas accidentales.

Además de `FaceMatchThreshold` y `UserMatchThreshold`, puede utilizar el atributo de respuesta `Similarity` como medio para reducir las identificaciones erróneas accidentales. Por ejemplo, puede optar por utilizar un umbral bajo (como el 80 %) para devolver más resultados. A continuación, puede utilizar el atributo de respuesta Similitud (porcentaje de similitud) para restringir la selección y filtrar por las respuestas correctas en la aplicación. Una vez más, usar una similitud superior (como, por ejemplo, del 99 % o más) reduce el riesgo de identificación errónea.

Guía para indexar rostros en escenarios comunes

A continuación se presentan las directrices para usar `IndexFaces` en situaciones comunes.

Aplicaciones críticas o de seguridad pública

- Llame a IndexFaces con imágenes que contengan un solo rostro por imagen y asocie el ID de rostro devuelto con el identificador del sujeto de la imagen.
- Puede utilizar DetectFaces antes de la indexación para comprobar que solo haya un rostro en la imagen. Si se detecta más de un rostro, vuelva a enviar la imagen después de revisarla y con solo

un rostro presente en ella. Esto impide indexar varios rostros y asociárselos a la misma persona involuntariamente.

Aplicaciones de uso compartido de fotografías y redes sociales

- Debe realizar una llamada a `IndexFaces` sin restricciones respecto a las imágenes que contienen varios rostros en casos de uso como los álbumes de familia. En estos casos, se debe identificar a cada persona en cada foto y utilizar esa información para agrupar las fotos en función de las personas que aparecen en ellas.

Uso general

- Indexe varias imágenes diferentes de la misma persona, especialmente con diferentes atributos faciales (posturas, vello, etc.), crear un usuario y asociar los distintos rostros a ese usuario para mejorar la calidad de los resultados coincidentes.
- Incluya un proceso de revisión que permita indexar las coincidencias erróneas con el identificador facial correcto, para mejorar la capacidad de encontrar coincidencias de rostros en lo sucesivo.
- Para obtener información sobre la calidad de las imágenes, consulte Recomendaciones para la comparación de rostros en las imágenes de entrada.

Búsqueda de rostros y usuarios dentro de una colección

Tras crear una colección de rostros y almacenar los vectores de and/or usuario de vectores de rostros, puede buscar coincidencias faciales en una colección de rostros. Con Amazon Rekognition puede buscar rostros en una colección que coincidan con:

- Un ID de rostro proporcionado (SearchFaces). Para obtener más información, consulte Búsqueda de un rostro con un ID de rostro.
- El rostro de mayor tamaño de una imagen proporcionada (SearchFacesByImage). Para obtener más información, consulte Búsqueda de un rostro con una imagen.
- Rostros en un vídeo almacenado. Para obtener más información, consulte Búsqueda de rostros en vídeos almacenados.
- Rostros en un vídeo en streaming. Para obtener más información, consulte Trabajar con eventos de vídeo en streaming.

Puede usar la operación `CompareFaces` para comparar un rostro de la imagen de origen con los rostros de la imagen de destino. El ámbito de esta comparación se limita a los rostros que se detectan en la imagen de destino. Para obtener más información, consulte Comparación de rostros en imágenes.

Las distintas operaciones de búsqueda que se muestran en la siguiente lista comparan un rostro (identificado mediante un `FaceId` o una imagen de entrada) con todos los rostros almacenados en una colección de rostros determinada:

- SearchFaces
- SearchFacesByImage
- SearchUsers
- SearchUsersByImage

Casos de uso para fines de seguridad pública

Si implementa sistemas de detección y comparación faciales en casos de uso relacionados con la seguridad pública, además de las recomendaciones de Prácticas recomendadas para sensores, vídeos e imágenes de entrada y Guía para indexar rostros en escenarios comunes, debe aplicar las prácticas recomendadas que se indican a continuación. En primer lugar, debe aplicar umbrales de confianza del 99 % o más con el fin de reducir la cantidad de errores y falsos positivos. En segundo lugar, debe incluir revisores humanos para que comprueben los resultados recibidos del sistema de detección o comparación de rostros y nunca deben tomarse decisiones basadas en los resultados del sistema sin que una persona los haya sometido previamente a revisión. Los sistemas de detección y comparación de rostros deben utilizarse como herramienta para filtrar los resultados y permitir que las personas revisen y estudien las opciones rápidamente. En tercer lugar, recomendamos transparencia respecto al uso de los sistemas de detección y comparación de rostros en estos casos de uso; esto incluye, siempre que sea posible, informar a los usuarios finales y a los sujetos del uso de estos sistemas, obtener su consentimiento para este tipo de uso y poner a su disposición un mecanismo que les permita aportar comentarios para mejorar el sistema.

Si usted es una organismo de las fuerzas del orden que utiliza la característica de comparación de rostros de Amazon Rekognition en relación con investigaciones criminales, debe cumplir los requisitos mostrados en las Condiciones de servicio de AWS. Esto incluye lo siguiente.

- Contar con personal debidamente formado que revise todas las decisiones para tomar medidas que puedan afectar a las libertades civiles de una persona o derechos humanos equivalentes.

- Formar al personal en el uso responsable de los sistemas de reconocimiento facial.
- Divulgar públicamente el uso que hace su organismo de los sistemas de reconocimiento facial.
- No utilizar Amazon Rekognition para la vigilancia constante de una persona sin revisión independiente o en circunstancias extremas.

En todos los casos, las coincidencias de la comparación facial deberían analizarse junto con otras pruebas de peso y no deberían tomarse como el único factor determinante para actuar. Sin embargo, si se utiliza la comparación facial en algunos non-law-enforcement escenarios (por ejemplo, para desbloquear un teléfono o autenticar la identidad de un empleado para acceder a un edificio de oficinas privado y seguro), estas decisiones no requerirían una auditoría manual porque no afectarían a las libertades civiles de una persona ni a sus derechos humanos equivalentes.

Si va a usar un sistema de detección o comparación de rostros para casos de uso relacionados con la seguridad pública, debe aplicar las prácticas recomendadas mencionadas anteriormente. Además, debe consultar los recursos publicados sobre el uso de la comparación facial. Esto incluye documentos como la Face Recognition Policy Template for State, Local, and Tribal Criminal Intelligence and Investigative Activities. Se trata de una plantilla de desarrollo de políticas de reconocimiento facial para su uso en actividades de inteligencia e investigación de delitos que está disponible a través de la Oficina de Asistencia Jurídica (Bureau of Justice Assistance) del Departamento de Justicia (Department of Justice) estadounidense. La plantilla proporciona varios recursos relativos a la comparación facial y a la identificación biométrica. Se ha diseñado para proporcionar a los organismos de seguridad pública, policiales y judiciales un marco para el desarrollo de políticas de comparación facial que cumpla la legislación aplicable, reduzca los riesgos para la privacidad y determine la responsabilidad y la vigilancia de las entidades. Otros recursos son las prácticas recomendadas para aplicar el reconocimiento facial a usos comerciales (Best Privacy Practices for Commercial Use of Facial Recognition) de la Administración Nacional de Telecomunicaciones e Información (National Telecommunications and Information Administration) y las mejores prácticas para los usos comunes del reconocimiento facial (Best Practices for Common Uses of Facial Recognition) elaboradas por el personal de la Comisión Federal de Comercio (Federal Trade Commission) (ambas estadounidenses). Es posible que se elaboren y publiquen otros recursos más adelante. Por ello, es importante mantenerse informado continuamente sobre este tema tan importante.

Le recordamos que debe cumplir todas las leyes aplicables cuando utilice los servicios de AWS y que no puede usar ningún servicio de AWS de ninguna manera que infrinja los derechos de otros o que pueda ser perjudicial para los demás. Esto significa que no puede usar los servicios de AWS en casos de uso con fines de seguridad pública de ningún modo que discrimine ilegalmente a personas

o que infrinja sus libertades civiles o los derechos procesales o de privacidad que le asisten. Debe obtener asesoramiento jurídico adecuado según sea necesario para revisar los requisitos y dudas legales relativos a su caso de uso.

Uso de Amazon Rekognition para mejorar la seguridad pública

Amazon Rekognition puede ayudar a la seguridad pública y al cumplimiento de la ley en situaciones como encontrar a niños perdidos, luchar contra la trata de seres humanos o prevenir delitos. En situaciones relacionadas con el cumplimiento de la ley y la seguridad pública, tenga en cuenta lo siguiente:

- Utilice Amazon Rekognition como el primer paso a la hora de encontrar posibles coincidencias. Gracias a las respuestas de las operaciones faciales de Amazon Rekognition, puede obtener rápidamente una serie de posibles coincidencias para examinarlas posteriormente.
- No utilice respuestas de Amazon Rekognition para tomar decisiones autónomas para escenarios que requieren el análisis de un humano. Si usted es un organismo de las fuerzas del orden que utiliza Amazon Rekognition para ayudar a identificar a una persona en relación con una investigación criminal y se van a tomar medidas basadas en la identificación que podrían afectar a las libertades civiles de esa persona o a derechos humanos equivalentes, la decisión de emprender medidas la debe tomar una persona debidamente formada en función de su juicio objetivo de las pruebas de identificación.
- Utilice un umbral de similitud del 99 % en los casos en los que sea necesario que las coincidencias de similitudes de los rostros sea muy precisa. Un ejemplo de esto sería la autenticación del acceso a un edificio.
- Cuando los derechos civiles son un problema, como en casos de uso relacionados con la ley, utilice umbrales de confianza del 99 % o superiores y asegúrese de que una persona revisa las predicciones de comparación facial para garantizar que no se infringen los derechos civiles de ningún individuo.
- Utilice un umbral de similitud inferior al 99 % en situaciones en las que resulte beneficioso contar con un conjunto más grande de coincidencias posibles. Un ejemplo de esto es la búsqueda de personas desaparecidas. En caso necesario, puede utilizar el atributo de respuesta Similarity para determinar la similitud de posibles coincidencias con la persona que desea reconocer.
- Tenga un plan para coincidencias de rostros falsos positivos devueltos por Amazon Rekognition. Por ejemplo, mejore las coincidencias utilizando varias imágenes de la misma persona cuando cree el índice con la operación IndexFaces. Para obtener más información, consulte Guía para indexar rostros en escenarios comunes.

En otros casos de uso (por ejemplo, las redes sociales), recomendamos aplicar su propio sentido común para evaluar si los resultados de Amazon Rekognition han de someterse a revisión humana. Además, en función de los requisitos de la aplicación, el umbral de similitud puede ser menor.

Creación de una colección

Puede utilizar la operación CreateCollection para crear una colección.

Para obtener más información, consulte Administración de una colección.

Para crear una colección (SDK)

1. Si aún no lo ha hecho:

   a. Cree o actualice un usuario de `AmazonRekognitionFullAccess` con permisos. Para obtener más información, consulte Paso 1: Configurar una cuenta de AWS y crear un usuario.

   b. Instale y configure el AWS CLI y el. AWS SDKs Para obtener más información, consulte Paso 2: Configure y AWS CLI AWS SDKs.

2. Consulte los siguientes ejemplos para llamar a la operación `CreateCollection`.

   Java

   En el siguiente ejemplo se crea una colección y se muestra el nombre de recurso de Amazon (ARN).

   Cambie el valor de `collectionId` por el nombre de la colección que desea crear.

```
//Copyright 2018 Amazon.com, Inc. or its affiliates. All Rights Reserved.
//PDX-License-Identifier: MIT-0 (For details, see https://github.com/awsdocs/
amazon-rekognition-developer-guide/blob/master/LICENSE-SAMPLECODE.)

package aws.example.rekognition.image;

import com.amazonaws.services.rekognition.AmazonRekognition;
import com.amazonaws.services.rekognition.AmazonRekognitionClientBuilder;
import com.amazonaws.services.rekognition.model.CreateCollectionRequest;
import com.amazonaws.services.rekognition.model.CreateCollectionResult;


public class CreateCollection {
```

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


      AmazonRekognition rekognitionClient =
 AmazonRekognitionClientBuilder.defaultClient();


      String collectionId = "MyCollection";
            System.out.println("Creating collection: " +
         collectionId );

        CreateCollectionRequest request = new CreateCollectionRequest()
                    .withCollectionId(collectionId);

      CreateCollectionResult createCollectionResult =
 rekognitionClient.createCollection(request);
      System.out.println("CollectionArn : " +
         createCollectionResult.getCollectionArn());
      System.out.println("Status code : " +
         createCollectionResult.getStatusCode().toString());

   }

}
```

Java V2

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

Sustituya el valor de `profile_name` en la línea que crea la sesión de Rekognition por el nombre de su perfil de desarrollador.

```
//snippet-start:[rekognition.java2.create_collection.import]
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import
 software.amazon.awssdk.services.rekognition.model.CreateCollectionResponse;
```

```
import
 software.amazon.awssdk.services.rekognition.model.CreateCollectionRequest;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
//snippet-end:[rekognition.java2.create_collection.import]

/**
* Before running this Java V2 code example, set up your development environment,
 including your credentials.
*
* For more information, see the following documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
*/
public class CreateCollection {

 public static void main(String[] args) {

     final String usage = "\n" +
         "Usage: " +
         "   <collectionName> \n\n" +
         "Where:\n" +
         "   collectionName - The name of the collection. \n\n";

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

     String collectionId = args[0];
     Region region = Region.US_EAST_1;
     RekognitionClient rekClient = RekognitionClient.builder()
         .region(region)
         .credentialsProvider(ProfileCredentialsProvider.create("profile-name"))
         .build();

     System.out.println("Creating collection: " +collectionId);
     createMyCollection(rekClient, collectionId );
     rekClient.close();
 }

 // snippet-start:[rekognition.java2.create_collection.main]
 public static void createMyCollection(RekognitionClient rekClient,String
 collectionId ) {
```

```
    try {
        CreateCollectionRequest collectionRequest =
CreateCollectionRequest.builder()
            .collectionId(collectionId)
            .build();

        CreateCollectionResponse collectionResponse =
rekClient.createCollection(collectionRequest);
        System.out.println("CollectionArn: " +
collectionResponse.collectionArn());
        System.out.println("Status code: " +
collectionResponse.statusCode().toString());

    } catch(RekognitionException e) {
        System.out.println(e.getMessage());
        System.exit(1);
    }
}
// snippet-end:[rekognition.java2.create_collection.main]
```

AWS CLI

Este AWS CLI comando muestra el resultado JSON de la operación `create-collection` CLI.

Reemplace el valor de `collection-id` por el nombre de la colección que desea crear.

Sustituya el valor de `profile_name` de por el nombre de su perfil de desarrollador.

```
aws rekognition create-collection --profile profile-name --collection-id
 "collection-name"
```

Python

En el siguiente ejemplo se crea una colección y se muestra el nombre de recurso de Amazon (ARN).

Cambie el valor de `collection_id` por el nombre de la colección que desea crear. Sustituya el valor de `profile_name` en la línea que crea la sesión de Rekognition por el nombre de su perfil de desarrollador.

```
# Copyright 2018 Amazon.com, Inc. or its affiliates. All Rights Reserved.
# PDX-License-Identifier: MIT-0 (For details, see https://github.com/awsdocs/
amazon-rekognition-developer-guide/blob/master/LICENSE-SAMPLECODE.)

import boto3

def create_collection(collection_id):
    session = boto3.Session(profile_name='profile-name')
    client = session.client('rekognition')

    # Create a collection
    print('Creating collection:' + collection_id)
    response = client.create_collection(CollectionId=collection_id)
    print('Collection ARN: ' + response['CollectionArn'])
    print('Status code: ' + str(response['StatusCode']))
    print('Done...')

def main():
    collection_id = "collection-id"
    create_collection(collection_id)

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

.NET

En el siguiente ejemplo se crea una colección y se muestra el nombre de recurso de Amazon (ARN).

Cambie el valor de `collectionId` por el nombre de la colección que desea crear.

```
//Copyright 2018 Amazon.com, Inc. or its affiliates. All Rights Reserved.
//PDX-License-Identifier: MIT-0 (For details, see https://github.com/awsdocs/
amazon-rekognition-developer-guide/blob/master/LICENSE-SAMPLECODE.)

using System;
using Amazon.Rekognition;
using Amazon.Rekognition.Model;

public class CreateCollection
{
    public static void Example()
```

```
    {
        AmazonRekognitionClient rekognitionClient = new
 AmazonRekognitionClient();

        String collectionId = "MyCollection";
        Console.WriteLine("Creating collection: " + collectionId);

        CreateCollectionRequest createCollectionRequest = new
 CreateCollectionRequest()
        {
            CollectionId = collectionId
        };

        CreateCollectionResponse createCollectionResponse =
 rekognitionClient.CreateCollection(createCollectionRequest);
        Console.WriteLine("CollectionArn : " +
 createCollectionResponse.CollectionArn);
        Console.WriteLine("Status code : " +
 createCollectionResponse.StatusCode);

    }
}
```

Node.JS

En el siguiente ejemplo, sustituya el valor de `region` por el nombre de la región asociada a su cuenta y sustituya el valor de `collectionName` por el nombre deseado de su colección.

Sustituya el valor de `profile_name` en la línea que crea la sesión de Rekognition por el nombre de su perfil de desarrollador.

```
//Copyright 2018 Amazon.com, Inc. or its affiliates. All Rights Reserved.
//PDX-License-Identifier: MIT-0 (For details, see https://github.com/awsdocs/
amazon-rekognition-developer-guide/blob/master/LICENSE-SAMPLECODE.)

import { CreateCollectionCommand} from  "@aws-sdk/client-rekognition";
import  { RekognitionClient } from "@aws-sdk/client-rekognition";
import {fromIni} from '@aws-sdk/credential-providers';

// Set the AWS Region.
const REGION = "region-name"; //e.g. "us-east-1"
// Set the profile name
const profileName = "profile-name"
```

```
// Name the collection
const collectionName = "collection-name"
const rekogClient = new RekognitionClient({region: REGION,
  credentials: fromIni({profile: profileName,}),
});

const createCollection = async (collectionName) => {
    try {
       console.log(`Creating collection: ${collectionName}`)
       const data = await rekogClient.send(new
 CreateCollectionCommand({CollectionId: collectionName}));
       console.log("Collection ARN:")
       console.log(data.CollectionARN)
       console.log("Status Code:")
       console.log(String(data.StatusCode))
       console.log("Success.",  data);
       return data;
    } catch (err) {
      console.log("Error", err.stack);
    }
  };

 createCollection(collectionName)
```

CreateCollection solicitud de operación

La entrada de `CreationCollection` es el nombre de la colección que desea crear.

```
{
    "CollectionId": "MyCollection"
}
```

CreateCollection respuesta de operación

Amazon Rekognition crea la colección y devuelve el Nombre de recurso de Amazon (ARN) de la colección que se acaba de crear.

```
{
   "CollectionArn": "aws:rekognition:us-east-1:acct-id:collection/examplecollection",
   "StatusCode": 200
}
```

Etiquetado de colecciones

Puede usar etiquetas para identificar, organizar, buscar y filtrar sus colecciones de Amazon Rekognition mediante etiquetas. Cada etiqueta es una marca que consta de una clave y un valor definidos por el usuario.

También puede utilizar etiquetas para controlar el acceso a una colección mediante la Administración de identidad y acceso (IAM). Para obtener más información, consulte Controlar el acceso a AWS los recursos mediante etiquetas de recursos.

Temas

- Agregar etiquetas a una colección nueva
- Agregar etiquetas a una colección existente
- Enumerar las etiquetas de una colección
- Eliminar etiquetas de una colección

Agregar etiquetas a una colección nueva

Puede agregar etiquetas a una colección a medida que la crea mediante la operación `CreateCollection`. Indique una o varias etiquetas en el parámetro de entrada `Tags` de la matriz.

AWS CLI

Sustituya el valor de `profile_name` de por el nombre de su perfil de desarrollador.

```
aws rekognition create-collection --collection-id "collection-name" --tags
 "{"key1":"value1","key2":"value2"}" --profile profile-name
```

Para los dispositivos de Windows:

```
aws rekognition create-collection --collection-id "collection-name" --tags
 "{\"key1\":\"value1\",\"key2\":\"value2\"}" --profile profile-name
```

Python

Sustituya el valor de `profile_name` en la línea que crea la sesión de Rekognition por el nombre de su perfil de desarrollador.

```
import boto3

def create_collection(collection_id):
    client = boto3.client('rekognition')

    # Create a collection
    print('Creating collection:' + collection_id)
    response = client.create_collection(CollectionId=collection_id)
    print('Collection ARN: ' + response['CollectionArn'])
    print('Status code: ' + str(response['StatusCode']))
    print('Done...')

def main():
    collection_id = 'NewCollectionName'
    create_collection(collection_id)

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

Agregar etiquetas a una colección existente

Para agregar una o varias etiquetas a una colección existente, utilice la operación `TagResource`. Indique el nombre de recurso de Amazon (ARN) de la colección (`ResourceArn`) y las etiquetas (`Tags`) que desea añadir. En el siguiente ejemplo se ve cómo añadir dos etiquetas.

AWS CLI

Sustituya el valor de `profile_name` de por el nombre de su perfil de desarrollador.

```
aws rekognition tag-resource --resource-arn collection-arn --tags
 "{"key1":"value1","key2":"value2"}" --profile profile-name
```

Para los dispositivos de Windows:

```
aws rekognition tag-resource --resource-arn collection-arn --tags "{\"key1\":
\"value1\",\"key2\":\"value2\"}" --profile profile-name
```

Python

Sustituya el valor de `profile_name` en la línea que crea la sesión de Rekognition por el nombre de su perfil de desarrollador.

```
# Copyright 2018 Amazon.com, Inc. or its affiliates. All Rights Reserved.
# PDX-License-Identifier: MIT-0 (For details, see https://github.com/awsdocs/amazon-rekognition-developer-guide/blob/master/LICENSE-SAMPLECODE.)

import boto3

def create_tag(collection_id):
    session = boto3.Session(profile_name='default')
    client = session.client('rekognition')
    response = client.tag_resource(ResourceArn=collection_id,
                                   Tags={
                                       "KeyName": "ValueName"
                                   })
    print(response)
    if "'HTTPStatusCode': 200" in str(response):
        print("Success!!")

def main():
    collection_arn = "collection-arn"
    create_tag(collection_arn)

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

> **Note**
>
> Si no conoce el nombre del recurso de Amazon de la colección, puede usar la operación `DescribeCollection`.

Enumerar las etiquetas de una colección

Para enumerar las etiquetas adjuntas a una colección, utilice la operación `ListTagsForResource` y especifique el ARN de la colección (`ResourceArn`). El resultado será la asignación de las claves y los valores de las etiquetas que se asocian a la colección especificada.

AWS CLI

Sustituya el valor de `profile_name` de por el nombre de su perfil de desarrollador.

```
aws rekognition list-tags-for-resource --resource-arn resource-arn --profile
 profile-name
```

Python

Sustituya el valor de `profile_name` en la línea que crea la sesión de Rekognition por el nombre de su perfil de desarrollador.

```
import boto3

def list_tags():
    client = boto3.client('rekognition')
    response =
 client.list_tags_for_resource(ResourceArn="arn:aws:rekognition:region-
name:5498347593847598:collection/NewCollectionName")
    print(response)

def main():
    list_tags()

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

El resultado muestra una lista de etiquetas adjuntas a la colección:

```
                {
    "Tags": {
        "Dept": "Engineering",
        "Name": "Ana Silva Carolina",
        "Role": "Developer"
    }
}
```

Eliminar etiquetas de una colección

Para eliminar una o más etiquetas de una colección, utilice la operación `UntagResource`. Indique el ARN del modelo (`ResourceArn`) y las claves de etiqueta (`Tag-Keys`) que desee eliminar.

AWS CLI

Sustituya el valor de `profile_name` de por el nombre de su perfil de desarrollador.

```
aws rekognition untag-resource --resource-arn resource-arn --profile profile-name --
tag-keys "key1" "key2"
```

Si lo prefiere, también puede indicar claves de etiqueta en este formato:

```
--tag-keys key1,key2
```

Python

Sustituya el valor de `profile_name` en la línea que crea la sesión de Rekognition por el nombre de su perfil de desarrollador.

```
import boto3

def list_tags():
    client = boto3.client('rekognition')
    response = client.untag_resource(ResourceArn="arn:aws:rekognition:region-
name:5498347593847598:collection/NewCollectionName", TagKeys=['KeyName'])
    print(response)

def main():
    list_tags()

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

Listado de colecciones

Puede utilizar la operación ListCollections para obtener una lista de las colecciones de la región que utiliza.

Para obtener más información, consulte Administración de una colección.

Para enumerar colecciones (SDK)

1. Si aún no lo ha hecho:

   a. Cree o actualice un usuario de `AmazonRekognitionFullAccess` con permisos. Para obtener más información, consulte Paso 1: Configurar una cuenta de AWS y crear un usuario.

   b. Instale y configure el AWS CLI y el AWS SDKs. Para obtener más información, consulte Paso 2: Configure y AWS CLI AWS SDKs.

2. Consulte los siguientes ejemplos para llamar a la operación `ListCollections`.

   Java

   En el ejemplo siguiente se muestra una lista de las colecciones de la región actual.

```
//Copyright 2018 Amazon.com, Inc. or its affiliates. All Rights Reserved.
//PDX-License-Identifier: MIT-0 (For details, see https://github.com/awsdocs/
amazon-rekognition-developer-guide/blob/master/LICENSE-SAMPLECODE.)

package aws.example.rekognition.image;

import java.util.List;
import com.amazonaws.services.rekognition.AmazonRekognition;
import com.amazonaws.services.rekognition.AmazonRekognitionClientBuilder;
import com.amazonaws.services.rekognition.model.ListCollectionsRequest;
import com.amazonaws.services.rekognition.model.ListCollectionsResult;

public class ListCollections {

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


      AmazonRekognition amazonRekognition =
 AmazonRekognitionClientBuilder.defaultClient();


      System.out.println("Listing collections");
      int limit = 10;
      ListCollectionsResult listCollectionsResult = null;
```

```
      String paginationToken = null;
      do {
         if (listCollectionsResult != null) {
            paginationToken = listCollectionsResult.getNextToken();
         }
         ListCollectionsRequest listCollectionsRequest = new
 ListCollectionsRequest()
                 .withMaxResults(limit)
                 .withNextToken(paginationToken);

 listCollectionsResult=amazonRekognition.listCollections(listCollectionsRequest);

         List < String > collectionIds =
 listCollectionsResult.getCollectionIds();
         for (String resultId: collectionIds) {
            System.out.println(resultId);
         }
      } while (listCollectionsResult != null &&
 listCollectionsResult.getNextToken() !=
         null);

   }
}
```

Java V2

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

```
//snippet-start:[rekognition.java2.list_collections.import]
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.ListCollectionsRequest;
import
 software.amazon.awssdk.services.rekognition.model.ListCollectionsResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import java.util.List;
//snippet-end:[rekognition.java2.list_collections.import]

/**
* Before running this Java V2 code example, set up your development environment,
 including your credentials.
```

```
*
* For more information, see the following documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
*/
public class ListCollections {

 public static void main(String[] args) {

     Region region = Region.US_EAST_1;
     RekognitionClient rekClient = RekognitionClient.builder()
         .region(region)
         .credentialsProvider(ProfileCredentialsProvider.create("profile-name"))
         .build();

     System.out.println("Listing collections");
     listAllCollections(rekClient);
     rekClient.close();
 }

 // snippet-start:[rekognition.java2.list_collections.main]
 public static void listAllCollections(RekognitionClient rekClient) {
     try {
         ListCollectionsRequest listCollectionsRequest =
 ListCollectionsRequest.builder()
             .maxResults(10)
             .build();

         ListCollectionsResponse response =
 rekClient.listCollections(listCollectionsRequest);
         List<String> collectionIds = response.collectionIds();
         for (String resultId : collectionIds) {
             System.out.println(resultId);
         }

     } catch (RekognitionException e) {
         System.out.println(e.getMessage());
         System.exit(1);
     }
 }
 // snippet-end:[rekognition.java2.list_collections.main]
}
```

AWS CLI

Este AWS CLI comando muestra el resultado JSON de la operación `list-collections` CLI. Sustituya el valor de `profile_name` de por el nombre de su perfil de desarrollador.

```
aws rekognition list-collections --profile profile-name
```

Python

En el ejemplo siguiente se muestra una lista de las colecciones de la región actual.

Sustituya el valor de `profile_name` en la línea que crea la sesión de Rekognition por el nombre de su perfil de desarrollador.

```
#Copyright 2018 Amazon.com, Inc. or its affiliates. All Rights Reserved.
#PDX-License-Identifier: MIT-0 (For details, see https://github.com/awsdocs/
amazon-rekognition-developer-guide/blob/master/LICENSE-SAMPLECODE.)

import boto3

def list_collections():

    max_results=2

    client=boto3.client('rekognition')

    #Display all the collections
    print('Displaying collections...')
    response=client.list_collections(MaxResults=max_results)
    collection_count=0
    done=False

    while done==False:
        collections=response['CollectionIds']

        for collection in collections:
            print (collection)
            collection_count+=1
        if 'NextToken' in response:
            nextToken=response['NextToken']

 response=client.list_collections(NextToken=nextToken,MaxResults=max_results)
```

```
        else:
            done=True

    return collection_count

def main():

    collection_count=list_collections()
    print("collections: " + str(collection_count))
if __name__ == "__main__":
    main()
```

.NET

En el ejemplo siguiente se muestra una lista de las colecciones de la región actual.

```
//Copyright 2018 Amazon.com, Inc. or its affiliates. All Rights Reserved.
//PDX-License-Identifier: MIT-0 (For details, see https://github.com/awsdocs/
amazon-rekognition-developer-guide/blob/master/LICENSE-SAMPLECODE.)

using System;
using Amazon.Rekognition;
using Amazon.Rekognition.Model;

public class ListCollections
{
    public static void Example()
    {
        AmazonRekognitionClient rekognitionClient = new
 AmazonRekognitionClient();

        Console.WriteLine("Listing collections");
        int limit = 10;

        ListCollectionsResponse listCollectionsResponse = null;
        String paginationToken = null;
        do
        {
            if (listCollectionsResponse != null)
                paginationToken = listCollectionsResponse.NextToken;
```

```
            ListCollectionsRequest listCollectionsRequest = new
 ListCollectionsRequest()
            {
                MaxResults = limit,
                NextToken = paginationToken
            };

            listCollectionsResponse =
 rekognitionClient.ListCollections(listCollectionsRequest);

            foreach (String resultId in listCollectionsResponse.CollectionIds)
                Console.WriteLine(resultId);
        } while (listCollectionsResponse != null &&
 listCollectionsResponse.NextToken != null);
    }
}
```

Node.js

Sustituya el valor de `profile_name` en la línea que crea la sesión de Rekognition por el nombre de su perfil de desarrollador.

```
//Copyright 2018 Amazon.com, Inc. or its affiliates. All Rights Reserved.
//PDX-License-Identifier: MIT-0 (For details, see https://github.com/awsdocs/
amazon-rekognition-developer-guide/blob/master/LICENSE-SAMPLECODE.)

import { ListCollectionsCommand } from  "@aws-sdk/client-rekognition";
import  { RekognitionClient } from "@aws-sdk/client-rekognition";
import {fromIni} from '@aws-sdk/credential-providers';

// Set the AWS Region.
const REGION = "region-name"; //e.g. "us-east-1"
// Set the profile name
const profileName = "profile-name"
// Name the collection
const rekogClient = new RekognitionClient({region: REGION,
  credentials: fromIni({profile: profileName,}),
});

const listCollection = async () => {
  var max_results = 10
  console.log("Displaying collections:")
```

```
  var response = await rekogClient.send(new ListCollectionsCommand({MaxResults:
 max_results}))
  var collection_count = 0
  var done = false
  while (done == false){
      var collections = response.CollectionIds
      collections.forEach(collection => {
          console.log(collection)
          collection_count += 1
      });
      return collection_count
  }
}

var collect_list = await listCollection()
console.log(collect_list)
```

ListCollections solicitud de operación

La entrada de `ListCollections` es el número máximo de colecciones que se va a devolver.

```
{
    "MaxResults": 2
}
```

Si la respuesta tiene más colecciones que los que requiere `MaxResults`, se devuelve un token que puede utilizar para obtener el siguiente conjunto de resultados, en una llamada posterior a `ListCollections`. Por ejemplo:

```
{
    "NextToken": "MGYZLAHX1T5a....",
    "MaxResults": 2
}
```

ListCollections respuesta de operación

Amazon Rekognition devuelve un conjunto de colecciones (`CollectionIds`). Una matriz independiente (`FaceModelVersions`) proporciona la versión del modelo de rostros utilizada para analizar los rostros de cada colección. Por ejemplo, en la siguiente respuesta JSON, la colección `MyCollection` analiza rostros mediante la versión 2.0 del modelo de rostros. La colección

`AnotherCollection` utiliza la versión 3.0 del modelo de rostros. Para obtener más información, consulte Descripción del control de versiones de los modelos.

`NextToken` es el token que se utiliza para obtener el siguiente conjunto de resultados, en una llamada posterior a `ListCollections`.

```
{
    "CollectionIds": [
        "MyCollection",
        "AnotherCollection"
    ],
    "FaceModelVersions": [
        "2.0",
        "3.0"
    ],
    "NextToken": "MGYZLAHX1T5a...."
}
```

Descripción de una colección

Puede utilizar la operación DescribeCollection para obtener la siguiente información acerca de una colección:

- El número de rostros que se indexan en la colección.
- La versión del modelo que se utiliza con la colección. Para obtener más información, consulte the section called “Descripción del control de versiones de los modelos”.
- El nombre de recurso de Amazon (ARN) de la colección.
- La fecha y hora de creación de la colección.

Para describir una colección (SDK)

1. Si aún no lo ha hecho:
   a. Cree o actualice un usuario de `AmazonRekognitionFullAccess` con permisos. Para obtener más información, consulte Paso 1: Configurar una cuenta de AWS y crear un usuario.
   b. Instale y configure el AWS CLI y el AWS SDKs. Para obtener más información, consulte Paso 2: Configure y AWS CLI AWS SDKs.

2. Consulte los siguientes ejemplos para llamar a la operación `DescribeCollection`.

   Java

   Este ejemplo describe una colección.

   Cambie el valor `collectionId` por el ID de la colección deseada.

```
//Copyright 2018 Amazon.com, Inc. or its affiliates. All Rights Reserved.
//PDX-License-Identifier: MIT-0 (For details, see https://github.com/awsdocs/
amazon-rekognition-developer-guide/blob/master/LICENSE-SAMPLECODE.)

package com.amazonaws.samples;

import com.amazonaws.services.rekognition.AmazonRekognition;
import com.amazonaws.services.rekognition.AmazonRekognitionClientBuilder;
import com.amazonaws.services.rekognition.model.DescribeCollectionRequest;
import com.amazonaws.services.rekognition.model.DescribeCollectionResult;


public class DescribeCollection {

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

      String collectionId = "CollectionID";

      AmazonRekognition rekognitionClient =
 AmazonRekognitionClientBuilder.defaultClient();


      System.out.println("Describing collection: " +
         collectionId );


        DescribeCollectionRequest request = new DescribeCollectionRequest()
                    .withCollectionId(collectionId);

      DescribeCollectionResult describeCollectionResult =
 rekognitionClient.describeCollection(request);
      System.out.println("Collection Arn : " +
         describeCollectionResult.getCollectionARN());
      System.out.println("Face count : " +
         describeCollectionResult.getFaceCount().toString());
```

```
      System.out.println("Face model version : " +
         describeCollectionResult.getFaceModelVersion());
      System.out.println("Created : " +
         describeCollectionResult.getCreationTimestamp().toString());

   }

}
```

Java V2

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

```
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import
 software.amazon.awssdk.services.rekognition.model.DescribeCollectionRequest;
import
 software.amazon.awssdk.services.rekognition.model.DescribeCollectionResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
//snippet-end:[rekognition.java2.describe_collection.import]

/**
* Before running this Java V2 code example, set up your development environment,
 including your credentials.
*
* For more information, see the following documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
*/
public class DescribeCollection {

 public static void main(String[] args) {

     final String usage = "\n" +
         "Usage: " +
         "   <collectionName>\n\n" +
         "Where:\n" +
```

```
        "   collectionName - The name of the Amazon Rekognition collection. \n
\n";

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

    String collectionName = args[0];
    Region region = Region.US_EAST_1;
    RekognitionClient rekClient = RekognitionClient.builder()
        .region(region)
        .credentialsProvider(ProfileCredentialsProvider.create("profile-name"))
        .build();

    describeColl(rekClient, collectionName);
    rekClient.close();
 }

 // snippet-start:[rekognition.java2.describe_collection.main]
 public static void describeColl(RekognitionClient rekClient, String
 collectionName) {

    try {
        DescribeCollectionRequest describeCollectionRequest =
 DescribeCollectionRequest.builder()
            .collectionId(collectionName)
            .build();

        DescribeCollectionResponse describeCollectionResponse =
 rekClient.describeCollection(describeCollectionRequest);
        System.out.println("Collection Arn : " +
 describeCollectionResponse.collectionARN());
        System.out.println("Created : " +
 describeCollectionResponse.creationTimestamp().toString());

    } catch(RekognitionException e) {
        System.out.println(e.getMessage());
        System.exit(1);
    }
 }
 // snippet-end:[rekognition.java2.describe_collection.main]
}
```

AWS CLI

Este AWS CLI comando muestra el resultado JSON de la operación `describe-collection` CLI. Cambie el valor de `collection-id` por el ID de la colección deseada. Sustituya el valor de `profile_name` en la línea que crea la sesión de Rekognition por el nombre de su perfil de desarrollador.

```
aws rekognition describe-collection --collection-id collection-name --profile
 profile-name
```

Python

Este ejemplo describe una colección.

Cambie el valor `collection_id` por el ID de la colección deseada. Sustituya el valor de `profile_name` en la línea que crea la sesión de Rekognition por el nombre de su perfil de desarrollador.

```
# Copyright 2018 Amazon.com, Inc. or its affiliates. All Rights Reserved.
# PDX-License-Identifier: MIT-0 (For details, see https://github.com/awsdocs/
amazon-rekognition-developer-guide/blob/master/LICENSE-SAMPLECODE.)

import boto3
from botocore.exceptions import ClientError

def describe_collection(collection_id):

    print('Attempting to describe collection ' + collection_id)

    session = boto3.Session(profile_name='default')
    client = session.client('rekognition')

    try:
    response = client.describe_collection(CollectionId=collection_id)
    print("Collection Arn: " + response['CollectionARN'])
    print("Face Count: " + str(response['FaceCount']))
    print("Face Model Version: " + response['FaceModelVersion'])
    print("Timestamp: " + str(response['CreationTimestamp']))

    except ClientError as e:
```

```
    if e.response['Error']['Code'] == 'ResourceNotFoundException':
    print('The collection ' + collection_id + ' was not found ')
    else:
    print('Error other than Not Found occurred: ' + e.response['Error']
['Message'])
    print('Done...')

def main():
    collection_id = 'collection-name'
    describe_collection(collection_id)

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

.NET

Este ejemplo describe una colección.

Cambie el valor `collectionId` por el ID de la colección deseada.

```
//Copyright 2018 Amazon.com, Inc. or its affiliates. All Rights Reserved.
//PDX-License-Identifier: MIT-0 (For details, see https://github.com/awsdocs/
amazon-rekognition-developer-guide/blob/master/LICENSE-SAMPLECODE.)

using System;
using Amazon.Rekognition;
using Amazon.Rekognition.Model;

public class DescribeCollection
{
    public static void Example()
    {
        AmazonRekognitionClient rekognitionClient = new
 AmazonRekognitionClient();

        String collectionId = "CollectionID";
        Console.WriteLine("Describing collection: " + collectionId);

        DescribeCollectionRequest describeCollectionRequest = new
 DescribeCollectionRequest()
        {
            CollectionId = collectionId
```

```
        };

        DescribeCollectionResponse describeCollectionResponse =
 rekognitionClient.DescribeCollection(describeCollectionRequest);
        Console.WriteLine("Collection ARN: " +
 describeCollectionResponse.CollectionARN);
        Console.WriteLine("Face count: " +
 describeCollectionResponse.FaceCount);
        Console.WriteLine("Face model version: " +
 describeCollectionResponse.FaceModelVersion);
        Console.WriteLine("Created: " +
 describeCollectionResponse.CreationTimestamp);
    }
}
```

Node.js

```
//Copyright 2018 Amazon.com, Inc. or its affiliates. All Rights Reserved.
//PDX-License-Identifier: MIT-0 (For details, see https://github.com/awsdocs/
amazon-rekognition-developer-guide/blob/master/LICENSE-SAMPLECODE.)

import { DescribeCollectionCommand } from  "@aws-sdk/client-rekognition";
import  { RekognitionClient } from "@aws-sdk/client-rekognition";
import {fromIni} from '@aws-sdk/credential-providers';

// Set the AWS Region.
const REGION = "region-name"; //e.g. "us-east-1"
// Set the profile name
const profileName = "profile-name"
// Name the collection
const rekogClient = new RekognitionClient({region: REGION,
  credentials: fromIni({profile: profileName,}),
});

// Name the collection
const collection_name = "collection-name"

const describeCollection = async (collectionName) => {
  try {
     console.log(`Attempting to describe collection named - ${collectionName}`)
     var response = await rekogClient.send(new
 DescribeCollectionCommand({CollectionId: collectionName}))
     console.log('Collection Arn:')
```

```
      console.log(response.CollectionARN)
      console.log('Face Count:')
      console.log(response.FaceCount)
      console.log('Face Model Version:')
      console.log(response.FaceModelVersion)
      console.log('Timestamp:')
      console.log(response.CreationTimestamp)
      return response; // For unit tests.
    } catch (err) {
      console.log("Error", err.stack);
    }
};

describeCollection(collection_name)
```

DescribeCollection solicitud de operación

La entrada de `DescribeCollection` es el ID de la colección deseada, tal y como se muestra en el siguiente ejemplo de JSON.

```
{
    "CollectionId": "MyCollection"
}
```

DescribeCollection respuesta de operación

La respuesta incluye:

- El número de rostros que se indexan en la colección, `FaceCount`.
- La versión del modelo facial que se utiliza con la colección, `FaceModelVersion`. Para obtener más información, consulte the section called “Descripción del control de versiones de los modelos”.
- La colección del Nombre de recurso de Amazon, `CollectionARN`.
- La fecha y hora de creación de la colección, `CreationTimestamp`. El valor de `CreationTimestamp` es el número de milisegundos desde el formato de tiempo Unix hasta la creación de la colección. El formato de tiempo Unix es 00:00:00 UTC (hora universal coordinada), jueves 1 de enero de 1970. Para obtener más información, consulte Unix Time (Tiempo Unix).

```
{
```

```
    "CollectionARN": "arn:aws:rekognition:us-east-1:nnnnnnnnnnnn:collection/
MyCollection",
    "CreationTimestamp": 1.533422155042E9,
    "FaceCount": 200,
    "UserCount" : 20,
    "FaceModelVersion": "1.0"
}
```

Eliminación de una colección

Puede utilizar la operación DeleteCollection para eliminar una colección.

Para obtener más información, consulte Administración de una colección.

Para eliminar una colección (SDK)

1. Si aún no lo ha hecho:
   a. Cree o actualice un usuario de `AmazonRekognitionFullAccess` con permisos. Para obtener más información, consulte Paso 1: Configurar una cuenta de AWS y crear un usuario.
   b. Instale y configure el AWS CLI y el AWS SDKs. Para obtener más información, consulte Paso 2: Configure y AWS CLI AWS SDKs.
2. Consulte los siguientes ejemplos para llamar a la operación `DeleteCollection`.

   Java

   Este ejemplo elimina una colección.

   Cambie el valor `collectionId` de la colección que desea eliminar.

```
//Copyright 2018 Amazon.com, Inc. or its affiliates. All Rights Reserved.
//PDX-License-Identifier: MIT-0 (For details, see https://github.com/awsdocs/
amazon-rekognition-developer-guide/blob/master/LICENSE-SAMPLECODE.)

package aws.example.rekognition.image;
import com.amazonaws.services.rekognition.AmazonRekognition;
import com.amazonaws.services.rekognition.AmazonRekognitionClientBuilder;
import com.amazonaws.services.rekognition.model.DeleteCollectionRequest;
import com.amazonaws.services.rekognition.model.DeleteCollectionResult;
```

```
public class DeleteCollection {

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

      AmazonRekognition rekognitionClient =
 AmazonRekognitionClientBuilder.defaultClient();

      String collectionId = "MyCollection";

      System.out.println("Deleting collections");

      DeleteCollectionRequest request = new DeleteCollectionRequest()
         .withCollectionId(collectionId);
      DeleteCollectionResult deleteCollectionResult =
 rekognitionClient.deleteCollection(request);

      System.out.println(collectionId + ": " +
 deleteCollectionResult.getStatusCode()
         .toString());

   }

}
```

Java V2

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

```
// snippet-start:[rekognition.java2.delete_collection.import]
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import
 software.amazon.awssdk.services.rekognition.model.DeleteCollectionRequest;
import
 software.amazon.awssdk.services.rekognition.model.DeleteCollectionResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
```

```
// snippet-end:[rekognition.java2.delete_collection.import]

/**
 * Before running this Java V2 code example, set up your development
 environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DeleteCollection {

    public static void main(String[] args) {

        final String usage = "\n" +
            "Usage: " +
            "   <collectionId> \n\n" +
            "Where:\n" +
            "   collectionId - The id of the collection to delete. \n\n";

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

        String collectionId = args[0];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
            .region(region)
            .credentialsProvider(ProfileCredentialsProvider.create("profile-
name"))
            .build();

        System.out.println("Deleting collection: " + collectionId);
        deleteMyCollection(rekClient, collectionId);
        rekClient.close();
    }

    // snippet-start:[rekognition.java2.delete_collection.main]
    public static void deleteMyCollection(RekognitionClient rekClient,String
 collectionId ) {

        try {
```

```
            DeleteCollectionRequest deleteCollectionRequest =
 DeleteCollectionRequest.builder()
                .collectionId(collectionId)
                .build();

            DeleteCollectionResponse deleteCollectionResponse =
 rekClient.deleteCollection(deleteCollectionRequest);
            System.out.println(collectionId + ": " +
 deleteCollectionResponse.statusCode().toString());

        } catch(RekognitionException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
    // snippet-end:[rekognition.java2.delete_collection.main]
}
```

AWS CLI

Este AWS CLI comando muestra el resultado JSON de la operación `delete-collection` CLI. Reemplace el valor de `collection-id` por el nombre de la colección que desea eliminar. Sustituya el valor de `profile_name` en la línea que crea la sesión de Rekognition por el nombre de su perfil de desarrollador.

```
aws rekognition delete-collection --collection-id collection-name --profile
 profile-name
```

Python

Este ejemplo elimina una colección.

Cambie el valor `collection_id` de la colección que desea eliminar. Sustituya el valor de `profile_name` en la línea que crea la sesión de Rekognition por el nombre de su perfil de desarrollador.

```
# Copyright 2018 Amazon.com, Inc. or its affiliates. All Rights Reserved.
# PDX-License-Identifier: MIT-0 (For details, see https://github.com/awsdocs/
amazon-rekognition-developer-guide/blob/master/LICENSE-SAMPLECODE.)

import boto3
from botocore.exceptions import ClientError
```

```
def delete_collection(collection_id):

    print('Attempting to delete collection ' + collection_id)
    session = boto3.Session(profile_name='default')
    client = session.client('rekognition')

    status_code = 0

    try:
        response = client.delete_collection(CollectionId=collection_id)
        status_code = response['StatusCode']

    except ClientError as e:
        if e.response['Error']['Code'] == 'ResourceNotFoundException':
            print('The collection ' + collection_id + ' was not found ')
        else:
            print('Error other than Not Found occurred: ' + e.response['Error']
['Message'])
        status_code = e.response['ResponseMetadata']['HTTPStatusCode']
    return (status_code)

def main():

    collection_id = 'collection-name'
    status_code = delete_collection(collection_id)
    print('Status code: ' + str(status_code))

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

.NET

Este ejemplo elimina una colección.

Cambie el valor `collectionId` de la colección que desea eliminar.

```
//Copyright 2018 Amazon.com, Inc. or its affiliates. All Rights Reserved.
//PDX-License-Identifier: MIT-0 (For details, see https://github.com/awsdocs/
amazon-rekognition-developer-guide/blob/master/LICENSE-SAMPLECODE.)

using System;
```

```
using Amazon.Rekognition;
using Amazon.Rekognition.Model;

public class DeleteCollection
{
    public static void Example()
    {
        AmazonRekognitionClient rekognitionClient = new
 AmazonRekognitionClient();

        String collectionId = "MyCollection";
        Console.WriteLine("Deleting collection: " + collectionId);

        DeleteCollectionRequest deleteCollectionRequest = new
 DeleteCollectionRequest()
        {
            CollectionId = collectionId
        };

        DeleteCollectionResponse deleteCollectionResponse =
 rekognitionClient.DeleteCollection(deleteCollectionRequest);
        Console.WriteLine(collectionId + ": " +
 deleteCollectionResponse.StatusCode);
    }
}
```

Node.js

Sustituya el valor de `profile_name` en la línea que crea la sesión de Rekognition por el nombre de su perfil de desarrollador.

```
//Copyright 2018 Amazon.com, Inc. or its affiliates. All Rights Reserved.
//PDX-License-Identifier: MIT-0 (For details, see https://github.com/awsdocs/
amazon-rekognition-developer-guide/blob/master/LICENSE-SAMPLECODE.)

import { DeleteCollectionCommand } from  "@aws-sdk/client-rekognition";
import  { RekognitionClient } from "@aws-sdk/client-rekognition";
import {fromIni} from '@aws-sdk/credential-providers';

// Set the AWS Region.
const REGION = "region-name"; //e.g. "us-east-1"
// Set the profile name
const profileName = "profile-name"
```

```
// Name the collection
const rekogClient = new RekognitionClient({region: REGION,
  credentials: fromIni({profile: profileName,}),
});

// Name the collection
const collection_name = "collection-name"

const deleteCollection = async (collectionName) => {
    try {
       console.log(`Attempting to delete collection named - ${collectionName}`)
       var response = await rekogClient.send(new
 DeleteCollectionCommand({CollectionId: collectionName}))
       var status_code = response.StatusCode
       if (status_code = 200){
           console.log("Collection successfully deleted.")
       }
       return response; // For unit tests.
    } catch (err) {
      console.log("Error", err.stack);
    }
  };

deleteCollection(collection_name)
```

DeleteCollection solicitud de operación

La entrada de `DeleteCollection` es el ID de la colección que se va a eliminar, tal y como se muestra en el siguiente ejemplo de JSON.

```
{
    "CollectionId": "MyCollection"
}
```

DeleteCollection respuesta de operación

La respuesta `DeleteCollection` contiene un código de estado HTTP que indica el éxito o el error de la operación. Se devuelve `200` si la colección se elimina correctamente.

```
{"StatusCode":200}
```

Agregar rostros a una colección

Puede utilizar la operación IndexFaces para detectar rostros en una imagen y añadirlas a una colección. Para cada rostro detectado, Amazon Rekognition extrae los rasgos faciales y almacena la información de rasgos en una base de datos. Además, el comando almacena los metadatos de cada uno de los rostros detectados en la colección de rostros especificada. Amazon Rekognition no conserva los bytes de las imágenes reales.

Para obtener información acerca de cómo proporcionar rostros adecuados para su indexación, consulte Recomendaciones para la comparación de rostros en las imágenes de entrada.

Para cada rostro, la operación `IndexFaces` conserva la siguiente información:

- Rasgos faciales multidimensionales: `IndexFaces` usa análisis faciales para extraer información multidimensional sobre los rasgos faciales y almacena la información en la colección de rostros. No se tiene acceso a esta información directamente. Sin embargo, Amazon Rekognition utiliza esta información cuando busca rostros coincidentes en una colección de rostros.

- Metadatos: los metadatos de cada rostro incluyen un cuadro delimitador, un nivel de confianza (que el cuadro delimitador contiene un rostro), IDs asignado por Amazon Rekognition (ID de rostro e ID de imagen) y un ID de imagen externo (si lo has proporcionado) en la solicitud. Esta información se devuelve en respuesta a la llamada a la API `IndexFaces`. Para ver un ejemplo, consulte el elemento `face` en la siguiente respuesta de ejemplo.

  El servicio devuelve estos metadatos en respuesta a las siguientes llamadas API:

  - `ListFaces`
  - Operaciones de búsqueda de rostros: las respuestas de SearchFaces y SearchFacesByImage devuelven la confianza de cada rostro coincidente, junto con los metadatos del rostro coincidente.

El número de rostros indexada por `IndexFaces` depende de la versión del modelo de detección de rostros que esté asociada a la colección de entrada. Para obtener más información, consulte Descripción del control de versiones de los modelos.

La información sobre los rostros indexados se devuelve en una matriz de objetos FaceRecord.

Es posible que desee asociar rostros indexados con la imagen en la que se detectaron. Por ejemplo, es posible que desee mantener un índice en el lado del cliente de imágenes y rostros en las imágenes. Para asociar rostros con una imagen, especifique un ID de imagen en el parámetro de solicitud `ExternalImageId`. El ID de imagen puede ser el nombre de archivo u otro ID que cree.

Además de la información anterior que la API conserva en la colección de rostros, la API devuelve también detalles del rostro que no se conservan en la colección. (Consulte el elemento `faceDetail` en la siguiente respuesta de ejemplo).

> **Note**
>
> `DetectFaces` devuelve la misma información, por lo que no es necesario llamar a `DetectFaces` y `IndexFaces` para la misma imagen.

Filtrado de rostros

La IndexFaces operación le permite filtrar los rostros indexados a partir de una imagen. Con `IndexFaces`, puede especificar el número máximo de rostros que desea indexar, o bien indicar que solo se indexen los rostros detectados con un índice de calidad alto.

Puede especificar el número máximo de rostros que se indexan mediante `IndexFaces` utilizando el parámetro de entrada `MaxFaces`. Esto resulta útil cuando se desea indexar los rostros de mayor tamaño de una imagen, pero no los más pequeños, como, por ejemplo, los de las personas que están de pie en segundo plano.

De forma predeterminada, `IndexFaces` selecciona un estándar de calidad que se utiliza para filtrar los rostros. Puede utilizar el parámetro de entrada `QualityFilter` para establecer explícitamente el estándar de calidad. Los valores son:

- `AUTO`: Amazon Rekognition elige el estándar de calidad que se usa para filtrar las caras (valor predeterminado).
- `LOW`: todos los rostros, excepto los de menor calidad, están indexados.
- `MEDIUM`
- `HIGH`: solo se indexan los rostros de mayor calidad.
- `NONE`: no se filtran rostros en función de la calidad.

`IndexFaces` filtra rostros basándose en lo siguiente:

- El rostro es demasiado pequeño en comparación con las dimensiones de la imagen.
- El rostro está demasiado borroso.
- La imagen es demasiado oscura.
- El rostro tiene una postura extrema.
- El rostro no tiene suficiente detalle para incluirse en la búsqueda de rostros.

> **Note**
>
> Para utilizar el filtrado según la calidad, necesita una colección asociada a la versión 3, o posterior, del modelo de rostros. Para obtener la versión del modelo de rostros asociado a una colección, llame a DescribeCollection.

La información sobre los rostros no indexados por `IndexFaces` se devuelve en una matriz de objetos UnindexedFace. La matriz `Reasons` contiene una lista de razones por las que un rostro no se ha indexado. Por ejemplo, el valor `EXCEEDS_MAX_FACES` significa que un rostro no se ha indexado porque ya se ha detectado el número de rostros especificado por `MaxFaces`.

Para obtener más información, consulte Administración de rostros en una colección.

Para agregar rostros a una colección (SDK)

1. Si aún no lo ha hecho:
   a. Cree o actualice un usuario con los permisos `AmazonRekognitionFullAccess` y `AmazonS3ReadOnlyAccess`. Para obtener más información, consulte Paso 1: Configurar una cuenta de AWS y crear un usuario.
   b. Instale y configure el AWS CLI y el AWS SDKs. Para obtener más información, consulte Paso 2: Configure y AWS CLI AWS SDKs.
2. Suba una imagen (que contenga uno o varios rostros) en su bucket de Amazon S3.

   Para ver las instrucciones, consulte Carga de objetos en Amazon S3 en la Guía del usuario de Amazon Simple Storage Service.
3. Consulte los siguientes ejemplos para llamar a la operación `IndexFaces`.

Java

Este ejemplo muestra los ID de rostro para los rostros añadidos a la colección.

Cambie el valor de `collectionId` por el nombre de la colección a la que desea agregar un rostro. Reemplace los valores de `bucket` y `photo` por los nombre del bucket de Amazon S3 y de la imagen que utilizó en el paso 2. El parámetro `.withMaxFaces(1)` reduce el número de rostros indexados a 1. Elimínelo o cambie su valor según sus necesidades.

```
//Copyright 2018 Amazon.com, Inc. or its affiliates. All Rights Reserved.
//PDX-License-Identifier: MIT-0 (For details, see https://github.com/awsdocs/
amazon-rekognition-developer-guide/blob/master/LICENSE-SAMPLECODE.)

package aws.example.rekognition.image;

import com.amazonaws.services.rekognition.AmazonRekognition;
import com.amazonaws.services.rekognition.AmazonRekognitionClientBuilder;
import com.amazonaws.services.rekognition.model.FaceRecord;
import com.amazonaws.services.rekognition.model.Image;
import com.amazonaws.services.rekognition.model.IndexFacesRequest;
import com.amazonaws.services.rekognition.model.IndexFacesResult;
import com.amazonaws.services.rekognition.model.QualityFilter;
import com.amazonaws.services.rekognition.model.S3Object;
import com.amazonaws.services.rekognition.model.UnindexedFace;
import java.util.List;

public class AddFacesToCollection {
    public static final String collectionId = "MyCollection";
    public static final String bucket = "bucket";
    public static final String photo = "input.jpg";

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

        AmazonRekognition rekognitionClient =
 AmazonRekognitionClientBuilder.defaultClient();

        Image image = new Image()
                .withS3Object(new S3Object()
                .withBucket(bucket)
                .withName(photo));

        IndexFacesRequest indexFacesRequest = new IndexFacesRequest()
```

```
                .withImage(image)
                .withQualityFilter(QualityFilter.AUTO)
                .withMaxFaces(1)
                .withCollectionId(collectionId)
                .withExternalImageId(photo)
                .withDetectionAttributes("DEFAULT");

        IndexFacesResult indexFacesResult =
 rekognitionClient.indexFaces(indexFacesRequest);

        System.out.println("Results for " + photo);
        System.out.println("Faces indexed:");
        List<FaceRecord> faceRecords = indexFacesResult.getFaceRecords();
        for (FaceRecord faceRecord : faceRecords) {
            System.out.println("  Face ID: " +
 faceRecord.getFace().getFaceId());
            System.out.println("  Location:" +
 faceRecord.getFaceDetail().getBoundingBox().toString());
        }

        List<UnindexedFace> unindexedFaces =
 indexFacesResult.getUnindexedFaces();
        System.out.println("Faces not indexed:");
        for (UnindexedFace unindexedFace : unindexedFaces) {
            System.out.println("  Location:" +
 unindexedFace.getFaceDetail().getBoundingBox().toString());
            System.out.println("  Reasons:");
            for (String reason : unindexedFace.getReasons()) {
                System.out.println("   " + reason);
            }
        }
    }
}
```

Java V2

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

```
//snippet-start:[rekognition.java2.add_faces_collection.import]
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
```

```
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.IndexFacesResponse;
import software.amazon.awssdk.services.rekognition.model.IndexFacesRequest;
import software.amazon.awssdk.services.rekognition.model.Image;
import software.amazon.awssdk.services.rekognition.model.QualityFilter;
import software.amazon.awssdk.services.rekognition.model.Attribute;
import software.amazon.awssdk.services.rekognition.model.FaceRecord;
import software.amazon.awssdk.services.rekognition.model.UnindexedFace;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import software.amazon.awssdk.services.rekognition.model.Reason;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;
import java.util.List;
//snippet-end:[rekognition.java2.add_faces_collection.import]

/**
* Before running this Java V2 code example, set up your development environment,
 including your credentials.
*
* For more information, see the following documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
*/
public class AddFacesToCollection {

 public static void main(String[] args) {

     final String usage = "\n" +
         "Usage: " +
         "    <collectionId> <sourceImage>\n\n" +
         "Where:\n" +
         "    collectionName - The name of the collection.\n" +
         "    sourceImage - The path to the image (for example, C:\\AWS\
\pic1.png). \n\n";

     if (args.length != 2) {
         System.out.println(usage);
         System.exit(1);
     }

     String collectionId = args[0];
     String sourceImage = args[1];
```

```
    Region region = Region.US_EAST_1;
    RekognitionClient rekClient = RekognitionClient.builder()
        .region(region)
        .credentialsProvider(ProfileCredentialsProvider.create("profile-name"))
        .build();

    addToCollection(rekClient, collectionId, sourceImage);
    rekClient.close();
}

// snippet-start:[rekognition.java2.add_faces_collection.main]
public static void addToCollection(RekognitionClient rekClient, String
collectionId, String sourceImage) {

    try {
        InputStream sourceStream = new FileInputStream(sourceImage);
        SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);
        Image souImage = Image.builder()
            .bytes(sourceBytes)
            .build();

        IndexFacesRequest facesRequest = IndexFacesRequest.builder()
            .collectionId(collectionId)
            .image(souImage)
            .maxFaces(1)
            .qualityFilter(QualityFilter.AUTO)
            .detectionAttributes(Attribute.DEFAULT)
            .build();

        IndexFacesResponse facesResponse = rekClient.indexFaces(facesRequest);
        System.out.println("Results for the image");
        System.out.println("\n Faces indexed:");
        List<FaceRecord> faceRecords = facesResponse.faceRecords();
        for (FaceRecord faceRecord : faceRecords) {
            System.out.println("  Face ID: " + faceRecord.face().faceId());
            System.out.println("  Location:" +
faceRecord.faceDetail().boundingBox().toString());
        }

        List<UnindexedFace> unindexedFaces = facesResponse.unindexedFaces();
        System.out.println("Faces not indexed:");
        for (UnindexedFace unindexedFace : unindexedFaces) {
            System.out.println("  Location:" +
unindexedFace.faceDetail().boundingBox().toString());
```

```
            System.out.println("  Reasons:");
            for (Reason reason : unindexedFace.reasons()) {
                System.out.println("Reason:  " + reason);
            }
        }

    } catch (RekognitionException | FileNotFoundException e) {
        System.out.println(e.getMessage());
        System.exit(1);
    }
 }
 // snippet-end:[rekognition.java2.add_faces_collection.main]
}
```

AWS CLI

Este AWS CLI comando muestra el resultado JSON de la operación `index-faces` CLI.

Reemplace el valor de `collection-id` por el nombre de la colección donde desea almacenar el rostro. Reemplace los valores `Bucket` y `Name` por el nombre del bucket de Amazon S3 y el nombre de archivo de imagen que utilizó en el paso 2. El parámetro `max-faces` reduce el número de rostros indexados a 1. Elimínelo o cambie su valor según sus necesidades. Sustituya el valor de `profile_name` en la línea que crea la sesión de Rekognition por el nombre de su perfil de desarrollador.

```
aws rekognition index-faces --image '{"S3Object":{"Bucket":"bucket-
name","Name":"file-name"}}' --collection-id "collection-id" \
                                --max-faces 1 --quality-filter "AUTO" --
detection-attributes "ALL" \
                                --external-image-id "example-image.jpg" --
profile profile-name
```

Si accede a la CLI en un dispositivo Windows, utilice comillas dobles en lugar de comillas simples y evite las comillas dobles internas con una barra invertida (es decir, \) para corregir cualquier error del analizador que pueda encontrar. Por ver un ejemplo, consulte lo siguiente:

```
aws rekognition index-faces --image "{\"S3Object\":{\"Bucket\":\"bucket-name\",
\"Name\":\"image-name\"}}" \
--collection-id "collection-id" --max-faces 1 --quality-filter "AUTO" --
detection-attributes "ALL" \
--external-image-id "example-image.jpg" --profile profile-name
```

Python

Este ejemplo muestra los ID de rostro para los rostros añadidos a la colección.

Cambie el valor de `collectionId` por el nombre de la colección a la que desea agregar un rostro. Reemplace los valores de `bucket` y `photo` por los nombre del bucket de Amazon S3 y de la imagen que utilizó en el paso 2. El parámetro de entrada `MaxFaces` determina el número de rostros indexados. Elimínelo o cambie su valor según sus necesidades. Sustituya el valor de `profile_name` en la línea que crea la sesión de Rekognition por el nombre de su perfil de desarrollador.

```
# Copyright 2018 Amazon.com, Inc. or its affiliates. All Rights Reserved.
# PDX-License-Identifier: MIT-0 (For details, see https://github.com/awsdocs/
amazon-rekognition-developer-guide/blob/master/LICENSE-SAMPLECODE.)

import boto3

def add_faces_to_collection(bucket, photo, collection_id):

    session = boto3.Session(profile_name='profile-name')
    client = session.client('rekognition')

    response = client.index_faces(CollectionId=collection_id,
                                  Image={'S3Object': {'Bucket': bucket, 'Name':
 photo}},
                                  ExternalImageId=photo,
                                  MaxFaces=1,
                                  QualityFilter="AUTO",
                                  DetectionAttributes=['ALL'])

    print('Results for ' + photo)
    print('Faces indexed:')
    for faceRecord in response['FaceRecords']:
        print('  Face ID: ' + faceRecord['Face']['FaceId'])
        print('  Location: {}'.format(faceRecord['Face']['BoundingBox']))

    print('Faces not indexed:')
    for unindexedFace in response['UnindexedFaces']:
        print(' Location: {}'.format(unindexedFace['FaceDetail']
['BoundingBox']))
```

```
        print(' Reasons:')
        for reason in unindexedFace['Reasons']:
            print('   ' + reason)
    return len(response['FaceRecords'])

def main():
    bucket = 'amzn-s3-demo-bucket'
    collection_id = 'collection-id'
    photo = 'photo-name'

    indexed_faces_count = add_faces_to_collection(bucket, photo, collection_id)
    print("Faces indexed count: " + str(indexed_faces_count))

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

.NET

Este ejemplo muestra los ID de rostro para los rostros añadidos a la colección.

Cambie el valor de `collectionId` por el nombre de la colección a la que desea agregar un rostro. Reemplace los valores de `bucket` y `photo` por los nombre del bucket de Amazon S3 y de la imagen que utilizó en el paso 2.

```
//Copyright 2018 Amazon.com, Inc. or its affiliates. All Rights Reserved.
//PDX-License-Identifier: MIT-0 (For details, see https://github.com/awsdocs/
amazon-rekognition-developer-guide/blob/master/LICENSE-SAMPLECODE.)

using System;
using System.Collections.Generic;
using Amazon.Rekognition;
using Amazon.Rekognition.Model;

public class AddFaces
{
    public static void Example()
    {
        String collectionId = "MyCollection";
        String bucket = "amzn-s3-demo-bucket";
        String photo = "input.jpg";

        AmazonRekognitionClient rekognitionClient = new
 AmazonRekognitionClient();
```

```
        Image image = new Image()
        {
            S3Object = new S3Object()
            {
                Bucket = bucket,
                Name = photo
            }
        };

        IndexFacesRequest indexFacesRequest = new IndexFacesRequest()
        {
            Image = image,
            CollectionId = collectionId,
            ExternalImageId = photo,
            DetectionAttributes = new List<String>(){ "ALL" }
        };

        IndexFacesResponse indexFacesResponse =
 rekognitionClient.IndexFaces(indexFacesRequest);

        Console.WriteLine(photo + " added");
        foreach (FaceRecord faceRecord in indexFacesResponse.FaceRecords)
            Console.WriteLine("Face detected: Faceid is " +
               faceRecord.Face.FaceId);
    }
}
```

IndexFaces solicitud de operación

La entrada de `IndexFaces` es la imagen que se va a indexar y la colección a la que se añadirá el rostro o los rostros.

```
{
    "CollectionId": "MyCollection",
    "Image": {
        "S3Object": {
            "Bucket": "bucket",
            "Name": "input.jpg"
        }
    },
    "ExternalImageId": "input.jpg",
```

```
    "DetectionAttributes": [
        "DEFAULT"
    ],
    "MaxFaces": 1,
    "QualityFilter": "AUTO"
}
```

IndexFaces respuesta de operación

`IndexFaces` devuelve información sobre los rostros que se han detectado en la imagen. Por ejemplo, la siguiente respuesta de JSON incluye los atributos de detección predeterminados para los rostros detectados en la imagen de entrada. El ejemplo también muestra rostros no indexados porque se ha superado el valor del parámetro de entrada `MaxFaces`: la matriz `Reasons` contiene EXCEEDS_MAX_FACES. Si un rostro no se indexa por razones de calidad, `Reasons` contiene valores como LOW_SHARPNESS o LOW_BRIGHTNESS. Para obtener más información, consulte UnindexedFace.

```
{
    "FaceModelVersion": "3.0",
    "FaceRecords": [
        {
            "Face": {
                "BoundingBox": {
                    "Height": 0.3247932195663452,
                    "Left": 0.5055555701255798,
                    "Top": 0.2743072211742401,
                    "Width": 0.21444444358348846
                },
                "Confidence": 99.99998474121094,
                "ExternalImageId": "input.jpg",
                "FaceId": "b86e2392-9da1-459b-af68-49118dc16f87",
                "ImageId": "09f43d92-02b6-5cea-8fbd-9f187db2050d"
            },
            "FaceDetail": {
                "BoundingBox": {
                    "Height": 0.3247932195663452,
                    "Left": 0.5055555701255798,
                    "Top": 0.2743072211742401,
                    "Width": 0.21444444358348846
                },
                "Confidence": 99.99998474121094,
                "Landmarks": [
```

```
                    {
                        "Type": "eyeLeft",
                        "X": 0.5751981735229492,
                        "Y": 0.4010535478591919
                    },
                    {
                        "Type": "eyeRight",
                        "X": 0.6511467099189758,
                        "Y": 0.4017036259174347
                    },
                    {
                        "Type": "nose",
                        "X": 0.6314528584480286,
                        "Y": 0.4710812568664551
                    },
                    {
                        "Type": "mouthLeft",
                        "X": 0.5879443287849426,
                        "Y": 0.5171778798103333
                    },
                    {
                        "Type": "mouthRight",
                        "X": 0.6444502472877502,
                        "Y": 0.5164633989334106
                    }
                ],
                "Pose": {
                    "Pitch": -10.313642501831055,
                    "Roll": -1.0316886901855469,
                    "Yaw": 18.079818725585938
                },
                "Quality": {
                    "Brightness": 71.2919921875,
                    "Sharpness": 78.74752044677734
                }
            }
        }
    ],
    "OrientationCorrection": "",
    "UnindexedFaces": [
        {
            "FaceDetail": {
                "BoundingBox": {
                    "Height": 0.1329464465379715,
```

```
                "Left": 0.5611110925674438,
                "Top": 0.6832437515258789,
                "Width": 0.08777777850627899
            },
            "Confidence": 92.37225341796875,
            "Landmarks": [
                {
                    "Type": "eyeLeft",
                    "X": 0.5796897411346436,
                    "Y": 0.7452847957611084
                },
                {
                    "Type": "eyeRight",
                    "X": 0.6078574657440186,
                    "Y": 0.742687463760376
                },
                {
                    "Type": "nose",
                    "X": 0.597953200340271,
                    "Y": 0.7620673179626465
                },
                {
                    "Type": "mouthLeft",
                    "X": 0.5884202122688293,
                    "Y": 0.7920381426811218
                },
                {
                    "Type": "mouthRight",
                    "X": 0.60627681016922,
                    "Y": 0.7919750809669495
                }
            ],
            "Pose": {
                "Pitch": 15.658954620361328,
                "Roll": -4.583454608917236,
                "Yaw": 10.558992385864258
            },
            "Quality": {
                "Brightness": 42.54612350463867,
                "Sharpness": 86.93206024169922
            }
        },
        "Reasons": [
            "EXCEEDS_MAX_FACES"
```

```
            ]
        }
    ]
}
```

Para obtener toda la información facial, especifique "ALL" para el parámetro de solicitud `DetectionAttributes`. Por ejemplo, en la siguiente respuesta de ejemplo, tenga en cuenta la información adicional del elemento `faceDetail`, que no se almacena de forma persistente en el servidor:

- 25 referencias faciales (en comparación con las cinco del ejemplo anterior)
- Diez atributos faciales (gafas, barba, oclusión, dirección de la mirada, etc.)
- Emociones (véase el elemento `emotion`)

El elemento `face` proporciona metadatos que se almacenan de forma persistente en el servidor.

`FaceModelVersion` es la versión del modelo de rostros asociado a la colección. Para obtener más información, consulte Descripción del control de versiones de los modelos.

`OrientationCorrection` es la orientación estimada de la imagen. No se devolverá información de corrección de la orientación si utiliza una versión del modelo de detección facial posterior a la versión 3. Para obtener más información, consulte Obtención de coordenadas de cuadro delimitador y orientación de imagen.

El siguiente ejemplo de respuesta muestra el JSON devuelto al especificar ["ALL"]:

```
{
    "FaceModelVersion": "3.0",
    "FaceRecords": [
        {
            "Face": {
                "BoundingBox": {
                    "Height": 0.06333333253860474,
                    "Left": 0.17185185849666595,
                    "Top": 0.7366666793823242,
                    "Width": 0.11061728745698929
                },
                "Confidence": 99.99999237060547,
                "ExternalImageId": "input.jpg",
                "FaceId": "578e2e1b-d0b0-493c-aa39-ba476a421a34",
                "ImageId": "9ba38e68-35b6-5509-9d2e-fcffa75d1653"
```

```
            },
            "FaceDetail": {
                "AgeRange": {
                    "High": 25,
                    "Low": 15
                },
                "Beard": {
                    "Confidence": 99.98077392578125,
                    "Value": false
                },
                "BoundingBox": {
                    "Height": 0.06333333253860474,
                    "Left": 0.17185185849666595,
                    "Top": 0.7366666793823242,
                    "Width": 0.11061728745698929
                },
                "Confidence": 99.99999237060547,
                "Emotions": [
                    {
                        "Confidence": 95.40877532958984,
                        "Type": "HAPPY"
                    },
                    {
                        "Confidence": 6.6088080406188965,
                        "Type": "CALM"
                    },
                    {
                        "Confidence": 0.7385611534118652,
                        "Type": "SAD"
                    }
                ],
                 "EyeDirection": {
                    "yaw": 16.299732,
                    "pitch": -6.407457,
                    "confidence": 99.968704
                              }
                "Eyeglasses": {
                    "Confidence": 99.96795654296875,
                    "Value": false
                },
                "EyesOpen": {
                    "Confidence": 64.0671157836914,
                    "Value": true
                },
```

```
                "Gender": {
                    "Confidence": 100,
                    "Value": "Female"
                },
                "Landmarks": [
                    {
                        "Type": "eyeLeft",
                        "X": 0.21361233294010162,
                        "Y": 0.757106363773346
                    },
                    {
                        "Type": "eyeRight",
                        "X": 0.2518567442893982,
                        "Y": 0.7599404454231262
                    },
                    {
                        "Type": "nose",
                        "X": 0.2262365221977234,
                        "Y": 0.7711842060089111
                    },
                    {
                        "Type": "mouthLeft",
                        "X": 0.2050037682056427,
                        "Y": 0.7801263332366943
                    },
                    {
                        "Type": "mouthRight",
                        "X": 0.2430567592382431,
                        "Y": 0.7836716771125793
                    },
                    {
                        "Type": "leftPupil",
                        "X": 0.2161938101053238,
                        "Y": 0.756662905216217
                    },
                    {
                        "Type": "rightPupil",
                        "X": 0.2523181438446045,
                        "Y": 0.7603650689125061
                    },
                    {
                        "Type": "leftEyeBrowLeft",
                        "X": 0.20066319406032562,
                        "Y": 0.7501518130302429
```

```
                    },
                    {
                        "Type": "leftEyeBrowUp",
                        "X": 0.2130996286869049,
                        "Y": 0.7480520606040955
                    },
                    {
                        "Type": "leftEyeBrowRight",
                        "X": 0.22584207355976105,
                        "Y": 0.7504606246948242
                    },
                    {
                        "Type": "rightEyeBrowLeft",
                        "X": 0.24509544670581818,
                        "Y": 0.7526801824569702
                    },
                    {
                        "Type": "rightEyeBrowUp",
                        "X": 0.2582615911960602,
                        "Y": 0.7516844868659973
                    },
                    {
                        "Type": "rightEyeBrowRight",
                        "X": 0.26881539821624756,
                        "Y": 0.7554477453231812
                    },
                    {
                        "Type": "leftEyeLeft",
                        "X": 0.20624476671218872,
                        "Y": 0.7568746209144592
                    },
                    {
                        "Type": "leftEyeRight",
                        "X": 0.22105035185813904,
                        "Y": 0.7582521438598633
                    },
                    {
                        "Type": "leftEyeUp",
                        "X": 0.21401576697826385,
                        "Y": 0.7553104162216187
                    },
                    {
                        "Type": "leftEyeDown",
                        "X": 0.21317370235919952,
```

```
                    "Y": 0.7584449648857117
                },
                {
                    "Type": "rightEyeLeft",
                    "X": 0.24393919110298157,
                    "Y": 0.7600628137588501
                },
                {
                    "Type": "rightEyeRight",
                    "X": 0.2598416209220886,
                    "Y": 0.7605880498886108
                },
                {
                    "Type": "rightEyeUp",
                    "X": 0.2519053518772125,
                    "Y": 0.7582084536552429
                },
                {
                    "Type": "rightEyeDown",
                    "X": 0.25177454948425293,
                    "Y": 0.7612871527671814
                },
                {
                    "Type": "noseLeft",
                    "X": 0.2185886949300766,
                    "Y": 0.774715781211853
                },
                {
                    "Type": "noseRight",
                    "X": 0.23328955471515656,
                    "Y": 0.7759330868721008
                },
                {
                    "Type": "mouthUp",
                    "X": 0.22446128726005554,
                    "Y": 0.7805567383766174
                },
                {
                    "Type": "mouthDown",
                    "X": 0.22087252140045166,
                    "Y": 0.7891407608985901
                }
            ],
            "MouthOpen": {
```

```
                    "Confidence": 95.87068939208984,
                    "Value": false
                },
                "Mustache": {
                    "Confidence": 99.9828109741211,
                    "Value": false
                },
                "Pose": {
                    "Pitch": -0.9409101605415344,
                    "Roll": 7.233824253082275,
                    "Yaw": -2.3602254390716553
                },
                "Quality": {
                    "Brightness": 32.01998519897461,
                    "Sharpness": 93.67259216308594
                },
                "Smile": {
                    "Confidence": 86.7142105102539,
                    "Value": true
                },
                "Sunglasses": {
                    "Confidence": 97.38925170898438,
                    "Value": false
                }
            }
        }
    ],
    "OrientationCorrection": "ROTATE_0"
    "UnindexedFaces": []
}
```


Enumerar rostros y usuarios asociados en una colección

Puede utilizar la ListFacesoperación para enumerar las caras y sus usuarios asociados en una colección.

Para obtener más información, consulte Administración de rostros en una colección.

Para mostrar una lista de los rostros de una colección (SDK)

1. Si aún no lo ha hecho:

a. Cree o actualice un usuario de `AmazonRekognitionFullAccess` con permisos. Para obtener más información, consulte Paso 1: Configurar una cuenta de AWS y crear un usuario.

b. Instale y configure el AWS CLI y el AWS SDKs. Para obtener más información, consulte Paso 2: Configure y AWS CLI AWS SDKs.

2. Consulte los siguientes ejemplos para llamar a la operación `ListFaces`.

Java

Este ejemplo muestra una lista de los rostros de una colección.

Cambie el valor de `collectionId` por el de la colección deseada.

```
//Copyright 2018 Amazon.com, Inc. or its affiliates. All Rights Reserved.
//PDX-License-Identifier: MIT-0 (For details, see https://github.com/awsdocs/
amazon-rekognition-developer-guide/blob/master/LICENSE-SAMPLECODE.)

package aws.example.rekognition.image;
import com.amazonaws.services.rekognition.AmazonRekognition;
import com.amazonaws.services.rekognition.AmazonRekognitionClientBuilder;
import com.amazonaws.services.rekognition.model.Face;
import com.amazonaws.services.rekognition.model.ListFacesRequest;
import com.amazonaws.services.rekognition.model.ListFacesResult;
import java.util.List;
import com.fasterxml.jackson.databind.ObjectMapper;



public class ListFacesInCollection {
    public static final String collectionId = "MyCollection";

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

      AmazonRekognition rekognitionClient =
 AmazonRekognitionClientBuilder.defaultClient();

      ObjectMapper objectMapper = new ObjectMapper();

      ListFacesResult listFacesResult = null;
      System.out.println("Faces in collection " + collectionId);
```

```
      String paginationToken = null;
      do {
         if (listFacesResult != null) {
            paginationToken = listFacesResult.getNextToken();
         }

         ListFacesRequest listFacesRequest = new ListFacesRequest()
                 .withCollectionId(collectionId)
                 .withMaxResults(1)
                 .withNextToken(paginationToken);

         listFacesResult =  rekognitionClient.listFaces(listFacesRequest);
         List < Face > faces = listFacesResult.getFaces();
         for (Face face: faces) {
            System.out.println(objectMapper.writerWithDefaultPrettyPrinter()
               .writeValueAsString(face));
         }
      } while (listFacesResult != null && listFacesResult.getNextToken() !=
         null);
   }

}
```

Java V2

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

```
// snippet-start:[rekognition.java2.list_faces_collection.import]
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.Face;
import software.amazon.awssdk.services.rekognition.model.ListFacesRequest;
import software.amazon.awssdk.services.rekognition.model.ListFacesResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import java.util.List;
// snippet-end:[rekognition.java2.list_faces_collection.import]

/**
 * Before running this Java V2 code example, set up your development
 environment, including your credentials.
```

```
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class ListFacesInCollection {

    public static void main(String[] args) {

        final String usage = "\n" +
            "Usage: " +
            "   <collectionId>\n\n" +
            "Where:\n" +
            "   collectionId - The name of the collection. \n\n";

        if (args.length < 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String collectionId = args[0];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
            .region(region)
            .credentialsProvider(ProfileCredentialsProvider.create("profile-
name"))
            .build();

        System.out.println("Faces in collection " + collectionId);
        listFacesCollection(rekClient, collectionId) ;
        rekClient.close();
    }

    // snippet-start:[rekognition.java2.list_faces_collection.main]
    public static void listFacesCollection(RekognitionClient rekClient, String
 collectionId ) {
        try {
            ListFacesRequest facesRequest = ListFacesRequest.builder()
                .collectionId(collectionId)
                .maxResults(10)
                .build();

            ListFacesResponse facesResponse = rekClient.listFaces(facesRequest);
```

```
            List<Face> faces = facesResponse.faces();
            for (Face face: faces) {
                System.out.println("Confidence level there is a face:
 "+face.confidence());
                System.out.println("The face Id value is "+face.faceId());
            }

        } catch (RekognitionException e) {
            System.out.println(e.getMessage());
            System.exit(1);
         }
      }
    // snippet-end:[rekognition.java2.list_faces_collection.main]
  }
```

AWS CLI

Este AWS CLI comando muestra el resultado JSON de la operación `list-faces` CLI. Reemplace el valor de `collection-id` por el nombre de la colección que desea enumerar. Sustituya el valor de `profile_name` en la línea que crea la sesión de Rekognition por el nombre de su perfil de desarrollador.

```
aws rekognition list-faces --collection-id "collection-id"  --profile profile-
name
```

Python

Este ejemplo muestra una lista de los rostros de una colección.

Sustituya el valor de `profile_name` en la línea que crea la sesión de Rekognition por el nombre de su perfil de desarrollador.

```
# Copyright 2018 Amazon.com, Inc. or its affiliates. All Rights Reserved.
# PDX-License-Identifier: MIT-0 (For details, see https://github.com/awsdocs/
amazon-rekognition-developer-guide/blob/master/LICENSE-SAMPLECODE.)

import boto3

def list_faces_in_collection(collection_id):
    maxResults = 2
    faces_count = 0
    tokens = True
```

```
    session = boto3.Session(profile_name='profile-name')
    client = session.client('rekognition')
    response = client.list_faces(CollectionId=collection_id,
                                 MaxResults=maxResults)

    print('Faces in collection ' + collection_id)

    while tokens:

        faces = response['Faces']

        for face in faces:
            print(face)
            faces_count += 1
        if 'NextToken' in response:
            nextToken = response['NextToken']
            response = client.list_faces(CollectionId=collection_id,
                                         NextToken=nextToken,
 MaxResults=maxResults)
        else:
            tokens = False
    return faces_count

def main():
    collection_id = 'collection-id'
    faces_count = list_faces_in_collection(collection_id)
    print("faces count: " + str(faces_count))

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

.NET

Este ejemplo muestra una lista de los rostros de una colección.

Cambie el valor de `collectionId` por el de la colección deseada.

```
//Copyright 2018 Amazon.com, Inc. or its affiliates. All Rights Reserved.
//PDX-License-Identifier: MIT-0 (For details, see https://github.com/awsdocs/
amazon-rekognition-developer-guide/blob/master/LICENSE-SAMPLECODE.)

using System;
```

```
using Amazon.Rekognition;
using Amazon.Rekognition.Model;

public class ListFaces
{
    public static void Example()
    {
        String collectionId = "MyCollection";

        AmazonRekognitionClient rekognitionClient = new
 AmazonRekognitionClient();

        ListFacesResponse listFacesResponse = null;
        Console.WriteLine("Faces in collection " + collectionId);

        String paginationToken = null;
        do
        {
            if (listFacesResponse != null)
                paginationToken = listFacesResponse.NextToken;

            ListFacesRequest listFacesRequest = new ListFacesRequest()
            {
                CollectionId = collectionId,
                MaxResults = 1,
                NextToken = paginationToken
            };

            listFacesResponse = rekognitionClient.ListFaces(listFacesRequest);
            foreach(Face face in listFacesResponse.Faces)
                Console.WriteLine(face.FaceId);
        } while (listFacesResponse != null && !
String.IsNullOrEmpty(listFacesResponse.NextToken));
    }
}
```

ListFaces solicitud de operación

La entrada `ListFaces` es el identificador de la colección para la que quieres enumerar las caras. `MaxResultses` el número máximo de caras que se pueden devolver. ListFaces también incluye una lista de rostros con los IDs que filtrar los resultados y un seudónimo que se proporciona para mostrar solo los rostros asociados al usuario en cuestión.

```
{
    "CollectionId": "MyCollection",
    "MaxResults": 1
}
```

Si la respuesta tiene más rostros que los que requiere `MaxResults`, se devuelve un token que puede utilizar para obtener el siguiente conjunto de resultados, en una llamada posterior a `ListFaces`. Por ejemplo:

```
{
    "CollectionId": "MyCollection",
    "NextToken": "sm+5ythT3aeEVIR4WA....",
    "MaxResults": 1
}
```

ListFaces respuesta de operación

La respuesta de `ListFaces` es información acerca de los metadatos de los rostros almacenados en la colección especificada.

- FaceModelVersion— La versión del modelo de rostro asociada a la colección. Para obtener más información, consulte Descripción del control de versiones de los modelos.
- Faces: información sobre los rostros de la colección. Esto incluye información acerca de BoundingBox, la confianza, los identificadores de imágenes y el ID de rostro. Para obtener más información, consulte Face.
- NextToken— El token que se utiliza para obtener el siguiente conjunto de resultados.

```
{
    "FaceModelVersion": "6.0",
    "Faces": [
        {
            "Confidence": 99.76940155029297,
            "IndexFacesModelVersion": "6.0",
            "UserId": "demoUser2",
            "ImageId": "56a0ca74-1c83-39dd-b363-051a64168a65",
            "BoundingBox": {
                "Width": 0.03177810087800026,
                "Top": 0.36568498611450195,
```

```
                "Left": 0.3453829884529114,
                "Height": 0.056759100407361984
            },
            "FaceId": "c92265d4-5f9c-43af-a58e-12be0ce02bc3"
        },
        {
            "BoundingBox": {
                "Width": 0.03254450112581253,
                "Top": 0.6080359816551208,
                "Left": 0.5160620212554932,
                "Height": 0.06347999721765518
            },
            "IndexFacesModelVersion": "6.0",
            "FaceId": "851cb847-dccc-4fea-9309-9f4805967855",
            "Confidence": 99.94369506835938,
            "ImageId": "a8aed589-ceec-35f7-9c04-82e0b546b024"
        },
        {
            "BoundingBox": {
                "Width": 0.03094629943370819,
                "Top": 0.4218429923057556,
                "Left": 0.6513839960098267,
                "Height": 0.05266290158033371
            },
            "IndexFacesModelVersion": "6.0",
            "FaceId": "c0eb3b65-24a0-41e1-b23a-1908b1aaeac1",
            "Confidence": 99.82969665527344,
            "ImageId": "56a0ca74-1c83-39dd-b363-051a64168a65"
        }
    ]
}
```


Eliminación de rostros de una colección

Puede utilizar la operación DeleteFaces para eliminar rostros de una colección. Para obtener más información, consulte Administración de rostros en una colección.

Eliminación de los rostros de una colección

1. Si aún no lo ha hecho:

a. Cree o actualice un usuario de `AmazonRekognitionFullAccess` con permisos. Para obtener más información, consulte Paso 1: Configurar una cuenta de AWS y crear un usuario.

b. Instale y configure el AWS CLI y el AWS SDKs. Para obtener más información, consulte Paso 2: Configure y AWS CLI AWS SDKs.

2. Consulte los siguientes ejemplos para llamar a la operación `DeleteFaces`.

Java

En este ejemplo se elimina un único rostro de una colección.

Cambie el valor de `collectionId` por el nombre de la colección que contiene el rostro que desea eliminar. Cambie el valor de `faces` por el ID del rostro que desea eliminar. Para eliminar varias caras, añada la cara IDs a la `faces` matriz.

```
//Copyright 2018 Amazon.com, Inc. or its affiliates. All Rights Reserved.
//PDX-License-Identifier: MIT-0 (For details, see https://github.com/awsdocs/
amazon-rekognition-developer-guide/blob/master/LICENSE-SAMPLECODE.)

package aws.example.rekognition.image;
import com.amazonaws.services.rekognition.AmazonRekognition;
import com.amazonaws.services.rekognition.AmazonRekognitionClientBuilder;
import com.amazonaws.services.rekognition.model.DeleteFacesRequest;
import com.amazonaws.services.rekognition.model.DeleteFacesResult;

import java.util.List;


public class DeleteFacesFromCollection {
   public static final String collectionId = "MyCollection";
   public static final String faces[] = {"xxxxxxxx-xxxx-xxxx-xxxx-
xxxxxxxxxxxx"};

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

      AmazonRekognition rekognitionClient =
 AmazonRekognitionClientBuilder.defaultClient();


      DeleteFacesRequest deleteFacesRequest = new DeleteFacesRequest()
              .withCollectionId(collectionId)
```

```
                .withFaceIds(faces);

        DeleteFacesResult
 deleteFacesResult=rekognitionClient.deleteFaces(deleteFacesRequest);


        List < String > faceRecords = deleteFacesResult.getDeletedFaces();
        System.out.println(Integer.toString(faceRecords.size()) + " face(s)
 deleted:");
        for (String face: faceRecords) {
           System.out.println("FaceID: " + face);
        }
     }
}
```

Java V2

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

```
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.DeleteFacesRequest;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
// snippet-end:[rekognition.java2.delete_faces_collection.import]

/**
 * Before running this Java V2 code example, set up your development
 environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DeleteFacesFromCollection {
```

```
    public static void main(String[] args) {

        final String usage = "\n" +
            "Usage: " +
            "   <collectionId> <faceId> \n\n" +
            "Where:\n" +
            "   collectionId - The id of the collection from which faces are
 deleted. \n\n" +
            "   faceId - The id of the face to delete. \n\n";

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

        String collectionId = args[0];
        String faceId = args[1];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
            .region(region)
            .credentialsProvider(ProfileCredentialsProvider.create("profile-
name"))
            .build();

        System.out.println("Deleting collection: " + collectionId);
        deleteFacesCollection(rekClient, collectionId, faceId);
        rekClient.close();
    }

    // snippet-start:[rekognition.java2.delete_faces_collection.main]
    public static void deleteFacesCollection(RekognitionClient rekClient,
                                             String collectionId,
                                             String faceId) {

        try {
            DeleteFacesRequest deleteFacesRequest = DeleteFacesRequest.builder()
                .collectionId(collectionId)
                .faceIds(faceId)
                .build();

            rekClient.deleteFaces(deleteFacesRequest);
            System.out.println("The face was deleted from the collection.");

        } catch(RekognitionException e) {
```

```
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
    // snippet-end:[rekognition.java2.delete_faces_collection.main]
}
```

AWS CLI

Este AWS CLI comando muestra el resultado JSON de la operación `delete-faces` CLI. Reemplace el valor de `collection-id` por el nombre de la colección que contiene el rostro que desea eliminar. Sustituya el valor de `face-ids` por una matriz de caras IDs que desee eliminar. Sustituya el valor de `profile_name` en la línea que crea la sesión de Rekognition por el nombre de su perfil de desarrollador.

```
aws rekognition delete-faces --collection-id "collection-id" --face-ids "faceid"
 --profile profile-name
```

Python

En este ejemplo se elimina un único rostro de una colección.

Cambie el valor de `collectionId` por el nombre de la colección que contiene el rostro que desea eliminar. Cambie el valor de `faces` por el ID del rostro que desea eliminar. Para eliminar varias caras, añada la cara IDs a la `faces` matriz. Sustituya el valor de `profile_name` en la línea que crea la sesión de Rekognition por el nombre de su perfil de desarrollador.

```
# Copyright 2018 Amazon.com, Inc. or its affiliates. All Rights Reserved.
# PDX-License-Identifier: MIT-0 (For details, see https://github.com/awsdocs/
amazon-rekognition-developer-guide/blob/master/LICENSE-SAMPLECODE.)

import boto3

def delete_faces_from_collection(collection_id, faces):

    session = boto3.Session(profile_name='profile-name')
    client = session.client('rekognition')
    response = client.delete_faces(CollectionId=collection_id,
                                   FaceIds=faces)
```

```
    print(str(len(response['DeletedFaces'])) + ' faces deleted:')
    for faceId in response['DeletedFaces']:
        print(faceId)
    return len(response['DeletedFaces'])


def main():
    collection_id = 'collection-id'
    faces = []
    faces.append("xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx")

    faces_count = delete_faces_from_collection(collection_id, faces)
    print("deleted faces count: " + str(faces_count))

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

.NET

En este ejemplo se elimina un único rostro de una colección.

Cambie el valor de `collectionId` por el nombre de la colección que contiene el rostro que desea eliminar. Cambie el valor de `faces` por el ID del rostro que desea eliminar. Para eliminar varias caras, añada la cara IDs a la `faces` lista.

```
//Copyright 2018 Amazon.com, Inc. or its affiliates. All Rights Reserved.
//PDX-License-Identifier: MIT-0 (For details, see https://github.com/awsdocs/
amazon-rekognition-developer-guide/blob/master/LICENSE-SAMPLECODE.)

using System;
using System.Collections.Generic;
using Amazon.Rekognition;
using Amazon.Rekognition.Model;

public class DeleteFaces
{
    public static void Example()
    {
        String collectionId = "MyCollection";
        List<String> faces = new List<String>() { "xxxxxxxx-xxxx-xxxx-xxxx-
xxxxxxxxxxxx" };
```

```
        AmazonRekognitionClient rekognitionClient = new
 AmazonRekognitionClient();

        DeleteFacesRequest deleteFacesRequest = new DeleteFacesRequest()
        {
            CollectionId = collectionId,
            FaceIds = faces
        };

        DeleteFacesResponse deleteFacesResponse =
 rekognitionClient.DeleteFaces(deleteFacesRequest);
        foreach (String face in deleteFacesResponse.DeletedFaces)
            Console.WriteLine("FaceID: " + face);
    }
}
```

DeleteFaces solicitud de operación

La entrada `DeleteFaces` es el identificador de la colección que contiene las caras y una matriz de caras IDs para las caras que se van a eliminar.

```
{
    "CollectionId": "MyCollection",
    "FaceIds": [
        "daf29cac-f910-41e9-851f-6eeb0e08f973"
    ]
}
```

DeleteFaces respuesta de operación

La `DeleteFaces` respuesta contiene una matriz de caras IDs para las caras que se eliminaron.

```
{
    "DeletedFaces": [
        "daf29cac-f910-41e9-851f-6eeb0e08f973"
    ]
}
```

Si la cara IDs proporcionada en la entrada está asociada actualmente a un usuario, se devolverá por motivos válidos. UnsuccessfulFaceDeletions

```
{
   "DeletedFaces": [
        "daf29cac-f910-41e9-851f-6eeb0e08f973"
    ],
     "UnsuccessfulFaceDeletions" : [
        {
           "FaceId" : "0b683aed-a0f1-48b2-9b5e-139e9cc2a757",
           "UserId" : "demoUser1",
           "Reason" : ["ASSOCIATED_TO_AN_EXISTING_USER"]
        }
     ]
}
```

Creación de un usuario

Puedes usar la CreateUseroperación para crear un nuevo usuario en una colección usando un seudónimo único que proporciones. A continuación, puede asociar varios rostros al usuario recién creado.

Para crear un usuario (SDK)

1. Si aún no lo ha hecho:

   a. Cree o actualice una cuenta de usuario de IAM con permisos de `AmazonRekognitionFullAccess`. Para obtener más información, consulte Paso 1: Configurar una cuenta de AWS y crear un usuario.

   b. Instale y configure el AWS CLI y el AWS SDKs. Para obtener más información, consulte Paso 2: Configure y AWS CLI AWS SDKs.

2. Consulte los siguientes ejemplos para llamar a la operación `CreateUser`.

   Java

   En este ejemplo de código Java se crea un usuario.

```
import com.amazonaws.services.rekognition.AmazonRekognition;
import com.amazonaws.services.rekognition.AmazonRekognitionClientBuilder;
import com.amazonaws.services.rekognition.model.CreateUserRequest;
import com.amazonaws.services.rekognition.model.CreateUserResult;
```

```
public class CreateUser {

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


        AmazonRekognition rekognitionClient =
 AmazonRekognitionClientBuilder.defaultClient();

        //Replace collectionId and userId with the name of the user that you
 want to create in that target collection.

        String collectionId = "MyCollection";
        String userId = "demoUser";
        System.out.println("Creating new user: " +
                userId);

        CreateUserRequest request = new CreateUserRequest()
                .withCollectionId(collectionId)
                .withUserId(userId);

        rekognitionClient.createUser(request);
    }

}
```

AWS CLI

Este AWS CLI comando crea un usuario mediante la operación `create-user` CLI.

```
aws rekognition create-user --user-id user-id --collection-id collection-name --
region region-name
--client-request-token request-token
```

Python

Este ejemplo de código de Python crea un usuario.

```
# Copyright 2018 Amazon.com, Inc. or its affiliates. All Rights Reserved.
# PDX-License-Identifier: MIT-0 (For details, see https://github.com/awsdocs/
amazon-rekognition-developer-guide/blob/master/LICENSE-SAMPLECODE.)

import boto3
from botocore.exceptions import ClientError
```

```
import logging

logger = logging.getLogger(__name__)
session = boto3.Session(profile_name='profile-name')
client = session.client('rekognition')

def create_user(collection_id, user_id):
    """
    Creates a new User within a collection specified by CollectionId.
    Takes UserId as a parameter, which is a user provided ID which
    should be unique within the collection.

    :param collection_id: The ID of the collection where the indexed faces will
 be stored at.
    :param user_id: ID for the UserID to be created. This ID needs to be unique
 within the collection.

    :return: The indexFaces response
    """
    try:
        logger.info(f'Creating user: {collection_id}, {user_id}')
        client.create_user(
            CollectionId=collection_id,
            UserId=user_id
        )
    except ClientError:
        logger.exception(f'Failed to create user with given user id:
 {user_id}')
        raise

def main():
    collection_id = "collection-id"
    user_id = "user-id"
    create_user(collection_id, user_id)

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

Go

Este ejemplo de código de Go usa el AWS Go SDK V2 y crea un usuario.

```
package main

import (
 "context"
 "fmt"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/service/rekognition"
)

func main() {
 // Load the AWS SDK configuration
 cfg, err := config.LoadDefaultConfig(context.TODO(), config.WithRegion("us-
west-2"))
 if err != nil {
  log.Fatalf("Failed to load configuration: %v", err)
 }

 // Create a Rekognition client
 client := rekognition.NewFromConfig(cfg)

 // Set up the input parameters
 input := &rekognition.CreateUserInput{
  CollectionId: aws.String("my-new-collection"), // Replace with your collection
 ID
  UserId:       aws.String("user12345678910"),   // Replace with desired user ID
 }

 // Call the CreateUser operation
 result, err := client.CreateUser(context.TODO(), input)
 if err != nil {
  log.Fatalf("Failed to create user: %v", err)
 }

 // Print out the results
 fmt.Printf("User created successfully:\n")

}
```

Eliminar un usuario

Puede utilizar la DeleteUseroperación para eliminar un usuario de una colección, en función del UserID proporcionado. Tenga en cuenta que todos los rostros asociados al UserID se disocian del UserID antes de que se elimine el UserID especificado.

Para eliminar un usuario (SDK)

1. Si aún no lo ha hecho:

   a. Cree o actualice una cuenta de usuario de IAM con permisos de `AmazonRekognitionFullAccess`. Para obtener más información, consulte Paso 1: Configurar una cuenta de AWS y crear un usuario.

   b. Instale y configure el AWS CLI y el. AWS SDKs Para obtener más información, consulte Paso 2: Configure y AWS CLI AWS SDKs.

2. Consulte los siguientes ejemplos para llamar a la operación `DeleteUser`.

   Java

   En este ejemplo de código de Java se elimina un usuario.

```
import com.amazonaws.services.rekognition.AmazonRekognition;
import com.amazonaws.services.rekognition.AmazonRekognitionClientBuilder;
import com.amazonaws.services.rekognition.model.DeleteUserRequest;
import com.amazonaws.services.rekognition.model.DeleteUserResult;


public class DeleteUser {

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


        AmazonRekognition rekognitionClient =
 AmazonRekognitionClientBuilder.defaultClient();

        //Replace collectionId and userId with the name of the user that you
 want to delete from that target collection.

        String collectionId = "MyCollection";
        String userId = "demoUser";
        System.out.println("Deleting existing user: " +
```

```
                userId);

        DeleteUserRequest request = new DeleteUserRequest()
                .withCollectionId(collectionId)
                .withUserId(userId);

        rekognitionClient.deleteUser(request);
    }

}
```

AWS CLI

Este AWS CLI comando elimina un usuario mediante la operación `create-user` CLI.

```
aws rekognition delete-user --collection-id MyCollection
--user-id user-id --collection-id collection-name --region region-name
```

Python

En este ejemplo de código de Python se elimina un usuario.

```
# Copyright 2018 Amazon.com, Inc. or its affiliates. All Rights Reserved.
# PDX-License-Identifier: MIT-0 (For details, see https://github.com/awsdocs/
amazon-rekognition-developer-guide/blob/master/LICENSE-SAMPLECODE.)

import boto3
from botocore.exceptions import ClientError
import logging

logger = logging.getLogger(__name__)
session = boto3.Session(profile_name='profile-name')
client = session.client('rekognition')

def delete_user(collection_id, user_id):
        """
        Delete the user from the given collection

        :param collection_id: The ID of the collection where user is stored.
        :param user_id: The ID of the user in the collection to delete.
        """
        logger.info(f'Deleting user: {collection_id}, {user_id}')
        try:
```

```
            client.delete_user(
                CollectionId=collection_id,
                UserId=user_id
            )
        except ClientError:
            logger.exception(f'Failed to delete user with given user id:
 {user_id}')
            raise

def main():
    collection_id = "collection-id"
    user_id = "user-id"
    delete_user(collection_id, user_id)

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

Asociar caras a un usuario

Puede utilizar la AssociateFacesoperación para asociar varias caras individuales a un solo usuario. Para asociar un rostro a un usuario, primero debe crear una colección y un usuario. Tenga en cuenta que los vectores faciales deben residir en la misma colección en la que reside el vector del usuario.

Para asociar rostros (SDK)

1. Si aún no lo ha hecho:

   a. Cree o actualice un usuario de `AmazonRekognitionFullAccess` con permisos. Para obtener más información, consulte Paso 1: Configurar una cuenta de AWS y crear un usuario.

   b. Instale y configure el AWS CLI y el AWS SDKs. Para obtener más información, consulte Paso 2: Configure y AWS CLI AWS SDKs.

2. Consulte los siguientes ejemplos para llamar a la operación `AssociateFaces`.

   Java

   En este ejemplo de código Java se asocia un rostro a un usuario.

```
import java.util.Arrays;
import java.util.List;
```

```
import com.amazonaws.services.rekognition.AmazonRekognition;
import com.amazonaws.services.rekognition.AmazonRekognitionClientBuilder;
import com.amazonaws.services.rekognition.model.AssociateFacesRequest;
import com.amazonaws.services.rekognition.model.AssociateFacesResult;


public class AssociateFaces {

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


        AmazonRekognition rekognitionClient =
 AmazonRekognitionClientBuilder.defaultClient();

        /* Replace the below configurations to allow you successfully run the
 example

           @collectionId: The collection where user and faces are stored
           @userId: The user which faces will get associated to
           @faceIds: The list of face IDs that will get associated to the given
 user
           @userMatchThreshold: Minimum User match confidence required for the
 face to
                                be associated with a User that has at least one
 faceID already associated
         */

        String collectionId = "MyCollection";
        String userId = "demoUser";
        String faceId1 = "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx";
        String faceId2 = "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx";
        List<String> faceIds = Arrays.asList(faceid1,faceid2);

        float userMatchThreshold = 0f;
        System.out.println("Associating faces to the existing user: " +
                userId);

        AssociateFacesRequest request = new AssociateFacesRequest()
                .withCollectionId(collectionId)
                .withUserId(userId)
                .withFaceIds(faceIds)
                .withUserMatchThreshold(userMatchThreshold);
```

```
        AssociateFacesResult result = rekognitionClient.associateFaces(request);

        System.out.println("Successful face associations: " +
 result.getAssociatedFaces().size());
        System.out.println("Unsuccessful face associations: " +
 result.getUnsuccessfulFaceAssociations().size());
    }

}
```

AWS CLI

Este AWS CLI comando asocia un rostro a un usuario mediante la operación `associate-faces` CLI.

```
aws rekognition associate-faces --user-id user-id --face-ids face-id-1 face-id-2
--collection-id collection-name
--region region-name
```

Python

En este ejemplo de código de Python se asocia un rostro a un usuario.

```
from botocore.exceptions import ClientError
import boto3
import logging

logger = logging.getLogger(__name__)
session = boto3.Session(profile_name='profile-name')
client = session.client('rekognition')

def associate_faces(collection_id, user_id, face_ids):
    """
    Associate stored faces within collection to the given user

    :param collection_id: The ID of the collection where user and faces are
 stored.
    :param user_id: The ID of the user that we want to associate faces to
    :param face_ids: The list of face IDs to be associated to the given user

    :return: response of AssociateFaces API
    """
```

```
    logger.info(f'Associating faces to user: {user_id}, {face_ids}')
    try:
        response = client.associate_faces(
            CollectionId=collection_id,
            UserId=user_id,
            FaceIds=face_ids
        )
        print(f'- associated {len(response["AssociatedFaces"])} faces')
    except ClientError:
        logger.exception("Failed to associate faces to the given user")
        raise
    else:
        print(response)
        return response

def main():
    face_ids = ["faceId1", "faceId2"]
    collection_id = "collection-id"
    user_id = "user-id"
    associate_faces(collection_id, user_id, face_ids)

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

AssociateFaces respuesta de operación

La respuesta para `AssociateFaces` incluye el `UserStatus`, que es el estado de la solicitud de disociación, así como una lista de los `FaceIds` que se van a asociar. También se devuelve una lista de `UnsuccessfulFaceAssociations`. Después de enviar una solicitud a `AssociateFaces`, la operación puede tardar aproximadamente un minuto en completarse.

Por este motivo, UserStatus se devuelve el, que puede tener los siguientes valores:

- CREADO: indica que el «usuario» se ha creado correctamente y que actualmente no tiene ningún rostro asociado. El «usuario» estará en este estado antes de que se realice una llamada AssociateFaces «» exitosa.
- ACTUALIZACIÓN: indica que el «usuario» se está actualizando para reflejar las associated/disassociated caras nuevas y pasará a ESTAR ACTIVO en unos segundos. Los resultados de la búsqueda pueden incluir la palabra «Usuario» en este estado y los clientes pueden optar por ignorarlo de los resultados devueltos.

- ACTIVO: indica que el «usuario» está actualizado para reflejar todas las associated/disassociated caras y se encuentra en un estado en el que se pueden realizar búsquedas.


```
{
    "UnsuccessfulFaceAssociations": [
        {
            "Reasons": [
                "LOW_MATCH_CONFIDENCE"
            ],
            "FaceId": "f5817d37-94f6-0000-bfee-1a2b3c4d5e6f",
            "Confidence": 0.9375374913215637
        },
        {
            "Reasons": [
                "ASSOCIATED_TO_A_DIFFERENT_IDENTITY"
            ],
            "FaceId": "851cb847-dccc-1111-bfee-1a2b3c4d5e6f",
            "UserId": "demoUser2"
        }
    ],
    "UserStatus": "UPDATING",
    "AssociatedFaces": [
        {
            "FaceId": "35ebbb41-7f67-2222-bfee-1a2b3c4d5e6f"
        }
    ]
}
```


Desasociar rostros de un usuario

Puede utilizar esta DisassociateFacesoperación para eliminar la asociación entre un seudónimo y un identificador facial.

Para desasociar rostros (SDK)

1. Si aún no lo ha hecho:

    a. Cree o actualice un usuario de `AmazonRekognitionFullAccess` con permisos. Para obtener más información, consulte Paso 1: Configurar una cuenta de AWS y crear un usuario.

b. Instale y configure el AWS CLI y el AWS SDKs. Para obtener más información, consulte Paso 2: Configure y AWS CLI AWS SDKs.

2. Consulte los siguientes ejemplos para llamar a la operación `DisassociateFaces`.

Java

En este ejemplo de Java, se elimina la asociación entre un FaceID y un UserID con la operación `DisassociateFaces`.

```
import java.util.Arrays;
import java.util.List;

import com.amazonaws.services.rekognition.AmazonRekognition;
import com.amazonaws.services.rekognition.AmazonRekognitionClientBuilder;
import com.amazonaws.services.rekognition.model.DisassociateFacesRequest;
import com.amazonaws.services.rekognition.model.DisassociateFacesResult;


public class DisassociateFaces {

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


        AmazonRekognition rekognitionClient =
 AmazonRekognitionClientBuilder.defaultClient();

        /* Replace the below configurations to allow you successfully run the
 example

           @collectionId: The collection where user and faces are stored
           @userId: The user which faces will get disassociated from
           @faceIds: The list of face IDs that will get disassociated from the
 given user
         */

        String collectionId = "MyCollection";
        String userId = "demoUser";
        String faceId1 = "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx";
        String faceId2 = "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx";
        List<String> faceIds = Arrays.asList(faceid1,faceid2);

        System.out.println("Disassociating faces from existing user: " +
```

```
                userId);

        DisassociateFacesRequest request = new DisassociateFacesRequest()
                .withCollectionId(collectionId)
                .withUserId(userId)
                .withFaceIds(faceIds)

        DisassociateFacesResult result =
 rekognitionClient.disassociateFaces(request);

        System.out.println("Successful face disassociations: " +
 result.getDisassociatedFaces().size());
        System.out.println("Unsuccessful face disassociations: " +
 result.getUnsuccessfulFaceDisassociations().size());
    }

}
```

AWS CLI

Este AWS CLI comando elimina la asociación entre un FaceID y un UserID con la operación `DisassociateFaces`

```
aws rekognition disassociate-faces --face-ids list-of-face-ids
--user-id user-id --collection-id collection-name --region region-name
```

Python

En el siguiente ejemplo, se elimina la asociación entre un FaceID y un UserID con la operación `DisassociateFaces`.

```
from botocore.exceptions import ClientError
import boto3
import logging

logger = logging.getLogger(__name__)
session = boto3.Session(profile_name='profile-name')
client = session.client('rekognition')

def disassociate_faces(collection_id, user_id, face_ids):
    """
    Disassociate stored faces within collection to the given user
```

```
    :param collection_id: The ID of the collection where user and faces are
 stored.
    :param user_id: The ID of the user that we want to disassociate faces from
    :param face_ids: The list of face IDs to be disassociated from the given
 user

    :return: response of AssociateFaces API
    """
    logger.info(f'Disssociating faces from user: {user_id}, {face_ids}')
    try:
        response = client.disassociate_faces(
            CollectionId=collection_id,
            UserId=user_id,
            FaceIds=face_ids
        )
        print(f'- disassociated {len(response["DisassociatedFaces"])} faces')
    except ClientError:
        logger.exception("Failed to disassociate faces from the given user")
        raise
    else:
        print(response)
        return response

def main():
    face_ids = ["faceId1", "faceId2"]
    collection_id = "collection-id"
    user_id = "user-id"
    disassociate_faces(collection_id, user_id, face_ids)

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

DisassociateFaces respuesta de operación

La respuesta para `DisassociateFaces` incluye el `UserStatus`, que es el estado de la solicitud de disociación, así como una lista de los `FaceIds` que se van a desasociar. También se devuelve una lista de `UnsuccessfulFaceDisassociations`. Tras enviar una solicitud a DisassociateFaces, la operación puede tardar aproximadamente un minuto en completarse. Por este motivo, UserStatus se devuelve el, que puede tener los siguientes valores:

- CREADO: indica que el «usuario» se ha creado correctamente y que actualmente no tiene ningún rostro asociado. El «usuario» estará en este estado antes de que se realice una llamada AssociateFaces «» exitosa.
- ACTUALIZACIÓN: indica que el «usuario» se está actualizando para reflejar las associated/disassociated caras nuevas y pasará a ESTAR ACTIVO en unos segundos. Los resultados de la búsqueda pueden incluir la palabra «Usuario» en este estado y los clientes pueden optar por ignorarlo de los resultados devueltos.
- ACTIVO: indica que el «usuario» está actualizado para reflejar todas las associated/disassociated caras y se encuentra en un estado en el que se pueden realizar búsquedas.

```
{
    "UserStatus": "UPDATING",
    "DisassociatedFaces": [
        {
            "FaceId": "c92265d4-5f9c-43af-a58e-12be0ce02bc3"
        }
    ],
    "UnsuccessfulFaceDisassociations": [
        {
            "Reasons": [
                "ASSOCIATED_TO_A_DIFFERENT_IDENTITY"
            ],
            "FaceId": "f5817d37-94f6-4335-bfee-6cf79a3d806e",
            "UserId": "demoUser1"
        }
    ]
}
```

Listado de usuarios en una colección

Puede utilizar la ListUsersoperación para enumerar UserIds y el. UserStatus Para ver las caras IDs asociadas a un UserID, utilice la operación. ListFaces

Para enumerar los usuarios (SDK)

1. Si aún no lo ha hecho:

a. Cree o actualice un usuario de `AmazonRekognitionFullAccess` con permisos. Para obtener más información, consulte Paso 1: Configurar una cuenta de AWS y crear un usuario.

b. Instale y configure el AWS CLI y el. AWS SDKs Para obtener más información, consulte Paso 2: Configure y AWS CLI AWS SDKs.

2. Consulte los siguientes ejemplos para llamar a la operación `ListUsers`.

Java

En este ejemplo de Java, se enumeran los usuarios de una colección mediante la operación `ListUsers`.

```
import java.util.List;
import com.amazonaws.services.rekognition.AmazonRekognition;
import com.amazonaws.services.rekognition.AmazonRekognitionClientBuilder;
import com.amazonaws.services.rekognition.model.ListUsersRequest;
import com.amazonaws.services.rekognition.model.ListUsersResult;
import com.amazonaws.services.rekognition.model.User;

public class ListUsers {

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


        AmazonRekognition amazonRekognition =
 AmazonRekognitionClientBuilder.defaultClient();


        System.out.println("Listing users");
        int limit = 10;
        ListUsersResult listUsersResult = null;
        String paginationToken = null;
        do {
            if (listUsersResult != null) {
                paginationToken = listUsersResult.getNextToken();
            }
            ListUsersRequest request = new ListUsersRequest()
                    .withCollectionId(collectionId)
                    .withMaxResults(limit)
                    .withNextToken(paginationToken);
            listUsersResult = amazonRekognition.listUsers(request);
```

```
            List<User> users = listUsersResult.getUsers();
            for (User currentUser: users) {
                System.out.println(currentUser.getUserId() + " : " +
 currentUser.getUserStatus());
            }
        } while (listUsersResult.getNextToken() != null);
    }
}
```

AWS CLI

Este AWS CLI comando muestra los usuarios de una colección con la `ListUsers` operación.

```
aws rekognition list-users --collection-id collection-id --max-results number-
of-max-results
```

Python

En el siguiente ejemplo, se muestran los usuarios de una colección con la operación `ListUsers`.

```
# Copyright 2018 Amazon.com, Inc. or its affiliates. All Rights Reserved.
# PDX-License-Identifier: MIT-0 (For details, see https://github.com/awsdocs/
amazon-rekognition-developer-guide/blob/master/LICENSE-SAMPLECODE.)

import boto3
from botocore.exceptions import ClientError
import logging
from pprint import pprint

logger = logging.getLogger(__name__)
session = boto3.Session(profile_name='profile-name')
client = session.client('rekognition')


def list_users(collection_id):
    """
    List all users from the given collection

    :param collection_id: The ID of the collection where user is stored.
```

```
    :return: response that contains list of Users found within given collection
    """
    logger.info(f'Listing the users in collection: {collection_id}')
    try:
        response = client.list_users(
            CollectionId=collection_id
        )
        pprint(response["Users"])
    except ClientError:
        logger.exception(f'Failed to list all user from given collection:
 {collection_id}')
        raise
    else:
        return response

def main():
    collection_id = "collection-id"
    list_users(collection_id)

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

ListUsers respuesta de operación

La respuesta a una solicitud ListUsers incluye una lista de los elementos `Users` de la colección junto con la `UsedId` dirección y `UserStatus` del usuario.

```
{
    "NextToken": "B1asJT3bAb/ttuGgPFV8BZoBZyGQzlUHXbuTNLh48a6enU7kXKw43hpOwizW7LOk/
Gk7Em09lznoq6+FcDCcSq2olrn7A98BLkt5keu+ZRVRUTyrXtT6J7Hmp
+ieQ2an6Zu0qzPfcdPeaJ9eAxG2d0WNrzJgi5hvmjoiSTTfKX3MQz1sduWQkvAAs4hZfhZoKFahFlqWofshCXa/
FHAAY3PL1PjxXbkNeSSMq8V7i1MlKCdrPVykCv9MokpPt7jtNvKPEZGUhxgBTFMxNWLEcFnzAiCWDg91dFy/
LalshPjXA9UVc5Gx9vIJNQ/
eO3cQRghAkCT3FOAiXsLAnA015ODTomZpWWVpqB21wKpI3LYmfAVFrDPGzpbTVlRmLsJm41bkmnBBBw9+DHz1Jn7zW
+qc5Fs3yaHu0f51Xg==",
    "Users": [
        {
            "UserId": "demoUser4",
            "UserStatus": "CREATED"
        },
```

```
        {
            "UserId": "demoUser2",
            "UserStatus": "CREATED"
        }
    ]
}
```

Búsqueda de un rostro con un ID de rostro

Puede utilizar la SearchFacesoperación para buscar usuarios en una colección que coincidan con la cara más grande de una imagen proporcionada.

El ID de rostro se devuelve en la respuesta de la operación IndexFaces cuando el rostro se detecta y se añade a una colección. Para obtener más información, consulte Administración de rostros en una colección.

Para buscar un rostro en una colección con su ID de rostro (SDK)

1. Si aún no lo ha hecho:
   a. Cree o actualice un usuario de `AmazonRekognitionFullAccess` con permisos. Para obtener más información, consulte Paso 1: Configurar una cuenta de AWS y crear un usuario.
   b. Instale y configure el AWS CLI y el AWS SDKs. Para obtener más información, consulte Paso 2: Configure y AWS CLI AWS SDKs.
2. Consulte los siguientes ejemplos para llamar a la operación `SearchFaces`.

   Java

   Este ejemplo muestra información acerca de los rostros que coinciden con un rostro identificado por su ID.

   Cambie el valor de `collectionID` por la colección que contiene el rostro requerido. Cambie el valor de `faceId` por el identificador del rostro que desea buscar.

```
//Copyright 2018 Amazon.com, Inc. or its affiliates. All Rights Reserved.
//PDX-License-Identifier: MIT-0 (For details, see https://github.com/awsdocs/
amazon-rekognition-developer-guide/blob/master/LICENSE-SAMPLECODE.)
```

```
package aws.example.rekognition.image;
import com.amazonaws.services.rekognition.AmazonRekognition;
import com.amazonaws.services.rekognition.AmazonRekognitionClientBuilder;
import com.fasterxml.jackson.databind.ObjectMapper;
import com.amazonaws.services.rekognition.model.FaceMatch;
import com.amazonaws.services.rekognition.model.SearchFacesRequest;
import com.amazonaws.services.rekognition.model.SearchFacesResult;
import java.util.List;


  public class SearchFaceMatchingIdCollection {
      public static final String collectionId = "MyCollection";
      public static final String faceId = "xxxxxxxx-xxxx-xxxx-xxxx-
xxxxxxxxxxxx";

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

        AmazonRekognition rekognitionClient =
 AmazonRekognitionClientBuilder.defaultClient();

        ObjectMapper objectMapper = new ObjectMapper();
      // Search collection for faces matching the face id.

      SearchFacesRequest searchFacesRequest = new SearchFacesRequest()
              .withCollectionId(collectionId)
              .withFaceId(faceId)
              .withFaceMatchThreshold(70F)
              .withMaxFaces(2);

       SearchFacesResult searchFacesByIdResult =
               rekognitionClient.searchFaces(searchFacesRequest);

       System.out.println("Face matching faceId " + faceId);
      List < FaceMatch > faceImageMatches =
 searchFacesByIdResult.getFaceMatches();
      for (FaceMatch face: faceImageMatches) {
         System.out.println(objectMapper.writerWithDefaultPrettyPrinter()
                 .writeValueAsString(face));

         System.out.println();
      }
    }
```

```
}
```

Ejecute el código de ejemplo. Se muestra información sobre rostros coincidentes.

Java V2

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

```
// snippet-start:[rekognition.java2.match_faces_collection.import]
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.SearchFacesRequest;
import software.amazon.awssdk.services.rekognition.model.SearchFacesResponse;
import software.amazon.awssdk.services.rekognition.model.FaceMatch;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import java.util.List;
// snippet-end:[rekognition.java2.match_faces_collection.import]

/**
 * Before running this Java V2 code example, set up your development
 environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class SearchFaceMatchingIdCollection {

    public static void main(String[] args) {

        final String usage = "\n" +
            "Usage: " +
            "   <collectionId> <sourceImage>\n\n" +
            "Where:\n" +
            "   collectionId - The id of the collection.  \n" +
            "   sourceImage - The path to the image (for example, C:\\AWS\
\pic1.png). \n\n";
```

```
        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String collectionId = args[0];
        String faceId = args[1];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
            .region(region)
            .credentialsProvider(ProfileCredentialsProvider.create("profile-
name"))
            .build();

        System.out.println("Searching for a face in a collections");
        searchFacebyId(rekClient, collectionId, faceId ) ;
        rekClient.close();
    }

    // snippet-start:[rekognition.java2.match_faces_collection.main]
    public static void searchFacebyId(RekognitionClient rekClient,String
 collectionId, String faceId) {

        try {
            SearchFacesRequest searchFacesRequest = SearchFacesRequest.builder()
                .collectionId(collectionId)
                .faceId(faceId)
                .faceMatchThreshold(70F)
                .maxFaces(2)
                .build();

            SearchFacesResponse imageResponse =
 rekClient.searchFaces(searchFacesRequest) ;
            System.out.println("Faces matching in the collection");
            List<FaceMatch> faceImageMatches = imageResponse.faceMatches();
            for (FaceMatch face: faceImageMatches) {
                System.out.println("The similarity level is
 "+face.similarity());
                System.out.println();
            }

        } catch (RekognitionException e) {
            System.out.println(e.getMessage());
            System.exit(1);
```

```
        }
    }
    // snippet-end:[rekognition.java2.match_faces_collection.main]
}
```

AWS CLI

Este AWS CLI comando muestra el resultado JSON de la operación `search-faces` CLI. Reemplace el valor de `face-id` por el identificador del rostro que desea buscar y reemplace el valor de `collection-id` por la colección en la que desea buscar. Sustituya el valor de `profile_name` en la línea que crea la sesión de Rekognition por el nombre de su perfil de desarrollador.

```
aws rekognition search-faces --face-id face-id --collection-id "collection-id"
 --profile profile-name
```

Python

Este ejemplo muestra información acerca de los rostros que coinciden con un rostro identificado por su ID.

Cambie el valor de `collectionID` por la colección que contiene el rostro requerido. Cambie el valor de `faceId` por el identificador del rostro que desea buscar. Sustituya el valor de `profile_name` en la línea que crea la sesión de Rekognition por el nombre de su perfil de desarrollador.

```
# Copyright 2018 Amazon.com, Inc. or its affiliates. All Rights Reserved.
# PDX-License-Identifier: MIT-0 (For details, see https://github.com/awsdocs/
amazon-rekognition-developer-guide/blob/master/LICENSE-SAMPLECODE.)

import boto3

def search_face_in_collection(face_id, collection_id):
    threshold = 90
    max_faces = 2

    session = boto3.Session(profile_name='profile-name')
    client = session.client('rekognition')

    response = client.search_faces(CollectionId=collection_id,
                                   FaceId=face_id,
```

```
                                    FaceMatchThreshold=threshold,
                                    MaxFaces=max_faces)

    face_matches = response['FaceMatches']
    print('Matching faces')
    for match in face_matches:
        print('FaceId:' + match['Face']['FaceId'])
        print('Similarity: ' + "{:.2f}".format(match['Similarity']) + "%")

    return len(face_matches)

def main():
    face_id = 'xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx'
    collection_id = 'collection-id'

    faces = []
    faces.append(face_id)

    faces_count = search_face_in_collection(face_id, collection_id)
    print("faces found: " + str(faces_count))

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

.NET

Este ejemplo muestra información acerca de los rostros que coinciden con un rostro identificado por su ID.

Cambie el valor de `collectionID` por la colección que contiene el rostro requerido. Cambie el valor de `faceId` por el identificador del rostro que desea buscar.

```
//Copyright 2018 Amazon.com, Inc. or its affiliates. All Rights Reserved.
//PDX-License-Identifier: MIT-0 (For details, see https://github.com/awsdocs/
amazon-rekognition-developer-guide/blob/master/LICENSE-SAMPLECODE.)

using System;
using Amazon.Rekognition;
using Amazon.Rekognition.Model;

public class SearchFacesMatchingId
{
    public static void Example()
```

```
    {
        String collectionId = "MyCollection";
        String faceId = "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx";

        AmazonRekognitionClient rekognitionClient = new
 AmazonRekognitionClient();

        // Search collection for faces matching the face id.

        SearchFacesRequest searchFacesRequest = new SearchFacesRequest()
        {
            CollectionId = collectionId,
            FaceId = faceId,
            FaceMatchThreshold = 70F,
            MaxFaces = 2
        };

        SearchFacesResponse searchFacesResponse =
 rekognitionClient.SearchFaces(searchFacesRequest);

        Console.WriteLine("Face matching faceId " + faceId);

        Console.WriteLine("Matche(s): ");
        foreach (FaceMatch face in searchFacesResponse.FaceMatches)
            Console.WriteLine("FaceId: " + face.Face.FaceId + ", Similarity: " +
 face.Similarity);
    }
}
```

Ejecute el código de ejemplo. Se muestra información sobre rostros coincidentes.

SearchFaces solicitud de operación

Dado un ID de rostro (cada rostro almacenado en una colección de rostros tiene un ID de rostro), `SearchFaces` busca rostros similares en la colección de rostros especificada. La respuesta no incluye el rostro que está buscando. Incluye solo rostros similares. De forma predeterminada, `SearchFaces` devuelve rostros para los que el algoritmo detecta una similitud superior al 80%. La similitud indica la exactitud con la que los rostros encontrados coinciden con el rostro de entrada. Si lo prefiere, puede utilizar `FaceMatchThreshold` para especificar un valor diferente.

```
{
```

```
    "CollectionId": "MyCollection",
    "FaceId": "0b683aed-a0f1-48b2-9b5e-139e9cc2a757",
    "MaxFaces": 2,
    "FaceMatchThreshold": 99
}
```

SearchFaces respuesta de operación

La operación devuelve una matriz de los rostros coincidentes encontrados y el ID del rostro que proporcionó como entrada.

```
{
    "SearchedFaceId": "7ecf8c19-5274-5917-9c91-1db9ae0449e2",
    "FaceMatches": [ list of face matches found ]
}
```

Para cada rostro coincidente encontrado, la respuesta incluye metadatos de similitud y de rostro, como se muestra en la siguiente respuesta de ejemplo:

```
{
   ...
    "FaceMatches": [
        {
            "Similarity": 100.0,
            "Face": {
                "BoundingBox": {
                    "Width": 0.6154,
                    "Top": 0.2442,
                    "Left": 0.1765,
                    "Height": 0.4692
                },
                "FaceId": "84de1c86-5059-53f2-a432-34ebb704615d",
                "Confidence": 99.9997,
                "ImageId": "d38ebf91-1a11-58fc-ba42-f978b3f32f60"
            }
        },
        {
            "Similarity": 84.6859,
            "Face": {
                "BoundingBox": {
                    "Width": 0.2044,
                    "Top": 0.2254,
```

```
                    "Left": 0.4622,
                    "Height": 0.3119
                },
                "FaceId": "6fc892c7-5739-50da-a0d7-80cc92c0ba54",
                "Confidence": 99.9981,
                "ImageId": "5d913eaf-cf7f-5e09-8c8f-cb1bdea8e6aa"
            }
        }
    ]
}
```

Búsqueda de un rostro con una imagen

Puede utilizar la operación SearchFacesByImage para buscar rostros en una colección que coincida con el rostro de mayor tamaño en una imagen suministrada.

Para obtener más información, consulte Búsqueda de rostros y usuarios dentro de una colección.

Para buscar un rostro en una colección con una imagen (SDK)

1. Si aún no lo ha hecho:

   a. Cree o actualice un usuario con los permisos `AmazonRekognitionFullAccess` y `AmazonS3ReadOnlyAccess`. Para obtener más información, consulte Paso 1: Configurar una cuenta de AWS y crear un usuario.

   b. Instale y configure el AWS CLI y el AWS SDKs. Para obtener más información, consulte Paso 2: Configure y AWS CLI AWS SDKs.

2. Suba una imagen (que contenga uno o varios rostros) en el bucket de S3.

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

3. Consulte los siguientes ejemplos para llamar a la operación `SearchFacesByImage`.

   Java

   Este ejemplo muestra información acerca de los rostros que coinciden con el rostro de mayor tamaño de una imagen. El ejemplo de código especifica los parámetros `FaceMatchThreshold` y `MaxFaces` para limitar los resultados que se devuelven en la respuesta.

En el siguiente ejemplo, cambie lo siguiente: el valor de `collectionId` por la colección en la que desea buscar; cambie el valor de búsqueda `bucket` por el bucket que contiene la imagen de entrada y cambie el valor de `photo` por la imagen de entrada.

```
//Copyright 2018 Amazon.com, Inc. or its affiliates. All Rights Reserved.
//PDX-License-Identifier: MIT-0 (For details, see https://github.com/awsdocs/
amazon-rekognition-developer-guide/blob/master/LICENSE-SAMPLECODE.)

package aws.example.rekognition.image;
import com.amazonaws.services.rekognition.AmazonRekognition;
import com.amazonaws.services.rekognition.AmazonRekognitionClientBuilder;
import com.amazonaws.services.rekognition.model.FaceMatch;
import com.amazonaws.services.rekognition.model.Image;
import com.amazonaws.services.rekognition.model.S3Object;
import com.amazonaws.services.rekognition.model.SearchFacesByImageRequest;
import com.amazonaws.services.rekognition.model.SearchFacesByImageResult;
import java.util.List;
import com.fasterxml.jackson.databind.ObjectMapper;


public class SearchFaceMatchingImageCollection {
    public static final String collectionId = "MyCollection";
    public static final String bucket = "bucket";
    public static final String photo = "input.jpg";

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

       AmazonRekognition rekognitionClient =
 AmazonRekognitionClientBuilder.defaultClient();

      ObjectMapper objectMapper = new ObjectMapper();

       // Get an image object from S3 bucket.
      Image image=new Image()
              .withS3Object(new S3Object()
                      .withBucket(bucket)
                      .withName(photo));

      // Search collection for faces similar to the largest face in the image.
      SearchFacesByImageRequest searchFacesByImageRequest = new
 SearchFacesByImageRequest()
              .withCollectionId(collectionId)
```

```
                .withImage(image)
                .withFaceMatchThreshold(70F)
                .withMaxFaces(2);

        SearchFacesByImageResult searchFacesByImageResult =
                rekognitionClient.searchFacesByImage(searchFacesByImageRequest);

        System.out.println("Faces matching largest face in image from" + photo);
       List < FaceMatch > faceImageMatches =
 searchFacesByImageResult.getFaceMatches();
       for (FaceMatch face: faceImageMatches) {
           System.out.println(objectMapper.writerWithDefaultPrettyPrinter()
                   .writeValueAsString(face));
          System.out.println();
       }
    }
}
```

Java V2

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

```
// snippet-start:[rekognition.java2.search_faces_collection.import]
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import
 software.amazon.awssdk.services.rekognition.model.SearchFacesByImageRequest;
import software.amazon.awssdk.services.rekognition.model.Image;
import
 software.amazon.awssdk.services.rekognition.model.SearchFacesByImageResponse;
import software.amazon.awssdk.services.rekognition.model.FaceMatch;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;
import java.util.List;
```

```
// snippet-end:[rekognition.java2.search_faces_collection.import]

/**
 * Before running this Java V2 code example, set up your development
 environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class SearchFaceMatchingImageCollection {

    public static void main(String[] args) {

        final String usage = "\n" +
            "Usage: " +
            "   <collectionId> <sourceImage>\n\n" +
            "Where:\n" +
            "   collectionId - The id of the collection.  \n" +
            "   sourceImage - The path to the image (for example, C:\\AWS\
\pic1.png). \n\n";

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String collectionId = args[0];
        String sourceImage = args[1];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
            .region(region)
            .credentialsProvider(ProfileCredentialsProvider.create("profile-
name"))
            .build();

        System.out.println("Searching for a face in a collections");
        searchFaceInCollection(rekClient, collectionId, sourceImage ) ;
        rekClient.close();
    }

    // snippet-start:[rekognition.java2.search_faces_collection.main]
```

```
   public static void searchFaceInCollection(RekognitionClient rekClient,String
 collectionId, String sourceImage) {

       try {
           InputStream sourceStream = new FileInputStream(new
 File(sourceImage));
           SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);
           Image souImage = Image.builder()
               .bytes(sourceBytes)
               .build();

           SearchFacesByImageRequest facesByImageRequest =
 SearchFacesByImageRequest.builder()
               .image(souImage)
               .maxFaces(10)
               .faceMatchThreshold(70F)
               .collectionId(collectionId)
               .build();

           SearchFacesByImageResponse imageResponse =
 rekClient.searchFacesByImage(facesByImageRequest) ;
           System.out.println("Faces matching in the collection");
           List<FaceMatch> faceImageMatches = imageResponse.faceMatches();
           for (FaceMatch face: faceImageMatches) {
               System.out.println("The similarity level is
 "+face.similarity());
               System.out.println();
           }

       } catch (RekognitionException | FileNotFoundException e) {
           System.out.println(e.getMessage());
           System.exit(1);
       }
   }
   // snippet-end:[rekognition.java2.search_faces_collection.main]
}
```

AWS CLI

Este AWS CLI comando muestra el resultado JSON de la operación `search-faces-by-image` CLI. Reemplace el valor de `Bucket` por el nombre del bucket de S3 que utilizó en el paso 2. Reemplace el valor de `Name` por el nombre de archivo de imagen que utilizó en el paso 2. Reemplace el valor de `collection-id` por la colección en la que desea buscar.

Sustituya el valor de `profile_name` en la línea que crea la sesión de Rekognition por el nombre de su perfil de desarrollador.

```
aws rekognition search-faces-by-image --image '{"S3Object":{"Bucket":"bucket-
name","Name":"image-name"}}' \
--collection-id "collection-id" --profile profile-name
```

Si accede a la CLI en un dispositivo Windows, utilice comillas dobles en lugar de comillas simples y evite las comillas dobles internas con una barra invertida (es decir, \) para corregir cualquier error del analizador que pueda encontrar. Por ver un ejemplo, consulte lo siguiente:

```
aws rekognition search-faces-by-image --image "{\"S3Object\":{\"Bucket\":
\"bucket-name\",\"Name\":\"image-name\"}}" \
--collection-id "collection-id" --profile profile-name
```

Python

Este ejemplo muestra información acerca de los rostros que coinciden con el rostro de mayor tamaño de una imagen. El ejemplo de código especifica los parámetros `FaceMatchThreshold` y `MaxFaces` para limitar los resultados que se devuelven en la respuesta.

En el siguiente ejemplo, cambie lo siguiente: el valor de `collectionId` por la colección que desea buscar y sustituya los valores de `bucket` y `photo` por los nombres del bucket de Amazon S3 y la imagen que utilizó en el Paso 2. Sustituya el valor de `profile_name` en la línea que crea la sesión de Rekognition por el nombre de su perfil de desarrollador.

```
#Copyright 2018 Amazon.com, Inc. or its affiliates. All Rights Reserved.
#PDX-License-Identifier: MIT-0 (For details, see https://github.com/awsdocs/
amazon-rekognition-developer-guide/blob/master/LICENSE-SAMPLECODE.)

import boto3

if __name__ == "__main__":

    bucket='amzn-s3-demo-bucket'
    collectionId='MyCollection'
```

```
    fileName='input.jpg'
    threshold = 70
    maxFaces=2

    client=boto3.client('rekognition')


    response=client.search_faces_by_image(CollectionId=collectionId,
                                Image={'S3Object':
{'Bucket':bucket,'Name':fileName}},
                                FaceMatchThreshold=threshold,
                                MaxFaces=maxFaces)


    faceMatches=response['FaceMatches']
    print ('Matching faces')
    for match in faceMatches:
            print ('FaceId:' + match['Face']['FaceId'])
            print ('Similarity: ' + "{:.2f}".format(match['Similarity']) + "%")
            print
```

.NET

Este ejemplo muestra información acerca de los rostros que coinciden con el rostro de mayor tamaño de una imagen. El ejemplo de código especifica los parámetros `FaceMatchThreshold` y `MaxFaces` para limitar los resultados que se devuelven en la respuesta.

En el siguiente ejemplo, cambie lo siguiente: el valor de `collectionId` por la colección que desea buscar y sustituya los valores de `bucket` y `photo` por los nombres del bucket de Amazon S3 y la imagen que utilizó en el paso 2.

```
//Copyright 2018 Amazon.com, Inc. or its affiliates. All Rights Reserved.
//PDX-License-Identifier: MIT-0 (For details, see https://github.com/awsdocs/
amazon-rekognition-developer-guide/blob/master/LICENSE-SAMPLECODE.)

using System;
using Amazon.Rekognition;
using Amazon.Rekognition.Model;

public class SearchFacesMatchingImage
```

```
{
    public static void Example()
    {
        String collectionId = "MyCollection";
        String bucket = "amzn-s3-demo-bucket";
        String photo = "input.jpg";

        AmazonRekognitionClient rekognitionClient = new
 AmazonRekognitionClient();

        // Get an image object from S3 bucket.
        Image image = new Image()
        {
            S3Object = new S3Object()
            {
                Bucket = bucket,
                Name = photo
            }
        };

        SearchFacesByImageRequest searchFacesByImageRequest = new
 SearchFacesByImageRequest()
        {
            CollectionId = collectionId,
            Image = image,
            FaceMatchThreshold = 70F,
            MaxFaces = 2
        };

        SearchFacesByImageResponse searchFacesByImageResponse =
 rekognitionClient.SearchFacesByImage(searchFacesByImageRequest);

        Console.WriteLine("Faces matching largest face in image from " + photo);
        foreach (FaceMatch face in searchFacesByImageResponse.FaceMatches)
            Console.WriteLine("FaceId: " + face.Face.FaceId + ", Similarity: " +
 face.Similarity);
    }
}
```

SearchFacesByImage solicitud de operación

Los parámetros de entrada de `SearchFacesImageByImage` son la colección en la que se busca la imagen y la ubicación de la imagen de origen. En este ejemplo, la imagen de origen se guarda en un bucket de Amazon S3 (`S3Object`). También se especifica el número máximo de rostros que se devuelven (`Maxfaces`) y la confianza mínima que debe asociarse para que se devuelva un rostro (`FaceMatchThreshold`).

```
{
    "CollectionId": "MyCollection",
    "Image": {
        "S3Object": {
            "Bucket": "bucket",
            "Name": "input.jpg"
        }
    },
    "MaxFaces": 2,
    "FaceMatchThreshold": 99
}
```

SearchFacesByImage respuesta de operación

Dada una imagen de entrada (.jpeg o .png), la operación detecta primero el rostro en la imagen de entrada y después busca rostros similares en la colección de rostros especificada.

> **Note**
>
> Si el servicio detecta varios rostros en la imagen de entrada, utiliza el rostro mayor detectado para buscarlo en la colección de rostros.

La operación devuelve una matriz de los rostros coincidentes encontrados y la información sobre el rostro de entrada. Esto incluye información como el cuadro de límite y el valor de confianza, que indica el nivel de confianza de que el cuadro contenga un rostro.

De forma predeterminada, `SearchFacesByImage` devuelve rostros para los que el algoritmo detecta una similitud superior al 80%. La similitud indica la exactitud con la que los rostros encontrados coinciden con el rostro de entrada. Si lo prefiere, puede utilizar `FaceMatchThreshold` para especificar un valor diferente. Para cada rostro coincidente encontrado, la respuesta incluye metadatos de similitud y de rostro, como se muestra en la siguiente respuesta de ejemplo:

```
{
    "FaceMatches": [
        {
            "Face": {
                "BoundingBox": {
                    "Height": 0.06333330273628235,
                    "Left": 0.1718519926071167,
                    "Top": 0.7366669774055481,
                    "Width": 0.11061699688434601
                },
                "Confidence": 100,
                "ExternalImageId": "input.jpg",
                "FaceId": "578e2e1b-d0b0-493c-aa39-ba476a421a34",
                "ImageId": "9ba38e68-35b6-5509-9d2e-fcffa75d1653"
            },
            "Similarity": 99.9764175415039
        }
    ],
    "FaceModelVersion": "3.0",
    "SearchedFaceBoundingBox": {
        "Height": 0.06333333253860474,
        "Left": 0.17185185849666595,
        "Top": 0.7366666793823242,
        "Width": 0.11061728745698929
    },
    "SearchedFaceConfidence": 99.99999237060547
}
```


Búsqueda de usuarios (ID de rostro/ID de usuario)

Puede utilizar la SearchUsersoperación para buscar usuarios en una colección específica que coincidan con un identificador facial o un identificador de usuario proporcionados. La operación muestra los resultados `UserIds` clasificados según la puntuación de similitud más alta por encima de la solicitada UserMatchThreshold. El ID de usuario se crea en la CreateUsers operación. Para obtener más información, consulte Administrar usuarios en una colección.

Para buscar usuarios (SDK)

1. Si aún no lo ha hecho:

a. Cree o actualice un usuario de `AmazonRekognitionFullAccess` con permisos. Para obtener más información, consulte Paso 1: Configurar una cuenta de AWS y crear un usuario.

b. Instale y configure el AWS CLI y el AWS SDKs. Para obtener más información, consulte Paso 2: Configure y AWS CLI AWS SDKs.

2. Consulte los siguientes ejemplos para llamar a la operación `SearchUsers`.

Java

En este ejemplo de Java, se buscan los usuarios de una colección mediante la operación `SearchUsers`.

```
import com.amazonaws.services.rekognition.AmazonRekognition;
import com.amazonaws.services.rekognition.AmazonRekognitionClientBuilder;
import com.amazonaws.services.rekognition.model.UserMatch;
import com.amazonaws.services.rekognition.model.SearchUsersRequest;
import com.amazonaws.services.rekognition.model.SearchUsersResult;
import com.amazonaws.services.rekognition.model.UserMatch;

public class SearchUsers {
    //Replace collectionId and faceId with the values you want to use.

    public static final String collectionId = "MyCollection";
    public static final String faceId = "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx";

    public static final String userd = 'demo-user';

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

        AmazonRekognition rekognitionClient =
 AmazonRekognitionClientBuilder.defaultClient();

        // Search collection for faces matching the user id.
        SearchUsersRequest request = new SearchUsersRequest()
                .withCollectionId(collectionId)
                .withUserId(userId);

        SearchUsersResult result =
                rekognitionClient.searchUsers(request);
```

```
        System.out.println("Printing first search result with matched user and
similarity score");
        for (UserMatch match: result.getUserMatches()) {
            System.out.println(match.getUser().getUserId() + " with similarity
score " + match.getSimilarity());
        }

        // Search collection for faces matching the face id.
        SearchUsersRequest request1 = new SearchUsersRequest()
                .withCollectionId(collectionId)
                .withFaceId(faceId);

        SearchUsersResult result1 =
                rekognitionClient.searchUsers(request1);

        System.out.println("Printing second search result with matched user and
similarity score");
        for (UserMatch match: result1.getUserMatches()) {
            System.out.println(match.getUser().getUserId() + " with similarity
score " + match.getSimilarity());
        }
    }
```

AWS CLI

Este AWS CLI comando busca los usuarios de una colección con la `SearchUsers` operación.

```
aws rekognition search-users --face-id face-id --collection-id collection-id --
region region-name
```

Python

En el siguiente ejemplo, se buscan los usuarios de una colección con la operación `SearchUsers`.

```
# Copyright 2018 Amazon.com, Inc. or its affiliates. All Rights Reserved.
# PDX-License-Identifier: MIT-0 (For details, see https://github.com/awsdocs/
amazon-rekognition-developer-guide/blob/master/LICENSE-SAMPLECODE.)

import boto3
from botocore.exceptions import ClientError
```

```
import logging

logger = logging.getLogger(__name__)
session = boto3.Session(profile_name='profile-name')
client = session.client('rekognition')

def search_users_by_face_id(collection_id, face_id):
    """
    SearchUsers operation with face ID provided as the search source

    :param collection_id: The ID of the collection where user and faces are
 stored.
    :param face_id: The ID of the face in the collection to search for.

    :return: response of SearchUsers API
    """
    logger.info(f'Searching for users using a face-id: {face_id}')
    try:
        response = client.search_users(
            CollectionId=collection_id,
            FaceId=face_id
        )
        print(f'- found {len(response["UserMatches"])} matches')
        print([f'- {x["User"]["UserId"]} - {x["Similarity"]}%' for x in
 response["UserMatches"]])
    except ClientError:
        logger.exception(f'Failed to perform SearchUsers with given face id:
 {face_id}')
        raise
    else:
        print(response)
        return response

def search_users_by_user_id(collection_id, user_id):
    """
    SearchUsers operation with user ID provided as the search source

    :param collection_id: The ID of the collection where user and faces are
 stored.
    :param user_id: The ID of the user in the collection to search for.

    :return: response of SearchUsers API
    """
    logger.info(f'Searching for users using a user-id: {user_id}')
```

```
    try:
        response = client.search_users(
            CollectionId=collection_id,
            UserId=user_id
        )
        print(f'- found {len(response["UserMatches"])} matches')
        print([f'- {x["User"]["UserId"]} - {x["Similarity"]}%' for x in
 response["UserMatches"]])
    except ClientError:
        logger.exception(f'Failed to perform SearchUsers with given face id:
 {user_id}')
        raise
    else:
        print(response)
        return response

def main():
    collection_id = "collection-id"
    user_id = "user-id"
    face_id = "face_id"
    search_users_by_face_id(collection_id, face_id)
    search_users_by_user_id(collection_id, user_id)

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

SearchUsers solicitud de operación

Con un FaceID o un UserID SearchUsers , busca coincidencias de usuario en el CollectionID especificado. De forma predeterminada, SearchUsers devuelve el nombre de usuario IDs para el que la puntuación de similitud es superior al 80%. La similitud indica en qué medida el UserID coincide con el FaceID o el UserID proporcionados. Si IDs se devuelven varios usuarios, se muestran en orden de mayor a menor puntuación de similitud. Si lo desea, puede utilizar el UserMatchThreshold para especificar un valor diferente. Para obtener más información, consulte Administrar usuarios en una colección.

A continuación se muestra un ejemplo de una SearchUsers solicitud que utiliza`UserId`:

```
{
   "CollectionId": "MyCollection",
```

```
   "UserId": "demoUser1",
   "MaxUsers": 2,
   "UserMatchThreshold": 99
}
```

El siguiente es un ejemplo de una SearchUsers solicitud que utiliza`FaceId`:

```
{
    "CollectionId": "MyCollection",
    "FaceId": "bff43c40-cfa7-4b94-bed8-8a08b2205107",
    "MaxUsers": 2,
    "UserMatchThreshold": 99
}
```

SearchUsers respuesta de operación

Si busca con un`FaceId`, la respuesta para SearchUsers incluye el `FaceId` para el`SearchedFace`, así como una lista de `UserMatches` y el `UserId` y `UserStatus` para cada usuario.

```
{
    "SearchedFace": {
        "FaceId": "bff43c40-cfa7-4b94-bed8-8a08b2205107"
    },
    "UserMatches": [
        {
            "User": {
                "UserId": "demoUser1",
                "UserStatus": "ACTIVE"
            },
            "Similarity": 100.0
        },
        {
            "User": {
                "UserId": "demoUser2",
                "UserStatus": "ACTIVE"
            },
            "Similarity": 99.97946166992188
        }
```

```
    ],
    "FaceModelVersion": "6"
}
```

Si busca con un`UserId`, la respuesta para SearchUsers incluye el `UserId` para`SearchedUser`, además de los demás elementos de respuesta.

```
{
    "SearchedUser": {
        "UserId": "demoUser1"
    },
    "UserMatches": [
        {
            "User": {
                "UserId": "demoUser2",
                "UserStatus": "ACTIVE"
            },
            "Similarity": 99.97946166992188
        }
    ],
    "FaceModelVersion": "6"
}
```

Búsqueda de usuarios (imagen)

`SearchUsersByImage` busca en el CollectionID especificado los usuarios de una colección que coincidan con el rostro más grande detectado en una imagen proporcionada. De forma predeterminada, SearchUsersByImage devuelve el nombre de usuario IDs para el que la puntuación de similitud es superior al 80%. La similitud indica en qué medida el UserID coincide con el rostro más grande detectado en la imagen suministrada. Si IDs se devuelven varios usuarios, se muestran en orden de mayor a menor puntuación de similitud. Si lo desea, puede utilizar el UserMatchThreshold para especificar un valor diferente. Para obtener más información, consulte the section called “Administrar usuarios en una colección”.

Para buscar usuarios por imagen (SDK)

1. Si aún no lo ha hecho:

   a. Cree o actualice un usuario de `AmazonRekognitionFullAccess` con permisos. Para obtener más información, consulte Paso 1: Configurar una cuenta de AWS y crear un usuario.

   b. Instale y configure el AWS CLI y el AWS SDKs. Para obtener más información, consulte Paso 2: Configure y AWS CLI AWS SDKs.

2. Consulte los siguientes ejemplos para llamar a la operación `SearchUsersByImage`.

   Java

   En este ejemplo de Java, se buscan los usuarios de una colección a partir de una imagen de entrada mediante la operación `SearchUsersByImage`.

```
import com.amazonaws.services.rekognition.AmazonRekognition;
import com.amazonaws.services.rekognition.AmazonRekognitionClientBuilder;
import com.amazonaws.services.rekognition.model.Image;
import com.amazonaws.services.rekognition.model.S3Object;
import com.amazonaws.services.rekognition.model.SearchUsersByImageRequest;
import com.amazonaws.services.rekognition.model.SearchUsersByImageResult;
import com.amazonaws.services.rekognition.model.UserMatch;


public class SearchUsersByImage {
    //Replace bucket, collectionId and photo with your values.
    public static final String collectionId = "MyCollection";
    public static final String s3Bucket = "bucket";
    public static final String s3PhotoFileKey = "input.jpg";

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

        AmazonRekognition rekognitionClient =
 AmazonRekognitionClientBuilder.defaultClient();


        // Get an image object from S3 bucket.
        Image image = new Image()
                .withS3Object(new S3Object()
                        .withBucket(s3Bucket)
                        .withName(s3PhotoFileKey));
```

```
        // Search collection for users similar to the largest face in the image.
        SearchUsersByImageRequest request = new SearchUsersByImageRequest()
                .withCollectionId(collectionId)
                .withImage(image)
                .withUserMatchThreshold(70F)
                .withMaxUsers(2);

        SearchUsersByImageResult result =
                rekognitionClient.searchUsersByImage(request);

        System.out.println("Printing search result with matched user and
 similarity score");
        for (UserMatch match: result.getUserMatches()) {
            System.out.println(match.getUser().getUserId() + " with similarity
 score " + match.getSimilarity());
        }
    }
}
```

AWS CLI

Este AWS CLI comando busca los usuarios de una colección en función de una imagen de entrada, con la `SearchUsersByImage` operación.

```
aws rekognition search-users-by-image --image '{"S3Object":
{"Bucket":"s3BucketName","Name":"file-name"}}' --collection-id MyCollectionId --
region region-name
```

Python

En el siguiente ejemplo, se buscan los usuarios de una colección a partir de una imagen de entrada mediante la operación `SearchUsersByImage`.

```
# Copyright 2018 Amazon.com, Inc. or its affiliates. All Rights Reserved.
# PDX-License-Identifier: MIT-0 (For details, see https://github.com/awsdocs/
amazon-rekognition-developer-guide/blob/master/LICENSE-SAMPLECODE.)

import boto3
from botocore.exceptions import ClientError
import logging
import os
```

```
logger = logging.getLogger(__name__)
session = boto3.Session(profile_name='profile-name')
client = session.client('rekognition')

def load_image(file_name):
    """
    helper function to load the image for indexFaces call from local disk

    :param image_file_name: The image file location that will be used by
 indexFaces call.
    :return: The Image in bytes
    """
    print(f'- loading image: {file_name}')
    with open(file_name, 'rb') as file:
        return {'Bytes': file.read()}

def search_users_by_image(collection_id, image_file):
    """
    SearchUsersByImage operation with user ID provided as the search source

    :param collection_id: The ID of the collection where user and faces are
 stored.
    :param image_file: The image that contains the reference face to search
 for.

    :return: response of SearchUsersByImage API
    """
    logger.info(f'Searching for users using an image: {image_file}')
    try:
        response = client.search_users_by_image(
            CollectionId=collection_id,
            Image=load_image(image_file)
        )
        print(f'- found {len(response["UserMatches"])} matches')
        print([f'- {x["User"]["UserId"]} - {x["Similarity"]}%' for x in
 response["UserMatches"]])
    except ClientError:
        logger.exception(f'Failed to perform SearchUsersByImage with given
 image: {image_file}')
        raise
    else:
        print(response)
        return response
```

```
def main():
    collection_id = "collection-id"
    IMAGE_SEARCH_SOURCE = os.getcwd() + '/image_path'
    search_users_by_image(collection_id, IMAGE_SEARCH_SOURCE)

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

SearchUsersByImage solicitud de operación

La solicitud para `SearchUsersByImage` incluye la colección en la que se busca la imagen y la ubicación de la imagen de origen. En este ejemplo, la imagen de origen se guarda en un bucket de Amazon S3 (`S3Object`). También se especifica el número máximo de usuarios que se devuelven (`MaxUsers`) y la confianza mínima que debe asociarse para que se devuelva un usuario (`UserMatchThreshold`).

```
{
    "CollectionId": "MyCollection",
    "Image": {
        "S3Object": {
            "Bucket": "bucket",
            "Name": "input.jpg"
        }
    },
    "MaxUsers": 2,
    "UserMatchThreshold": 99
}
```

SearchUsersByImage respuesta de operación

La respuesta para `SearchUsersByImage` incluye un `FaceDetail` objeto para`SearchedFace`, así como una lista de UserMatches con `UserIdSimilarity`, y `UserStatus` para cada uno de ellos. Si la imagen de entrada contenía más de una cara, también se UnsearchedFaces mostrará una lista de.

```
{
    "SearchedFace": {
```

```
        "FaceDetail": {
            "BoundingBox": {
                "Width": 0.23692893981933594,
                "Top": 0.19235000014305115,
                "Left": 0.39177176356315613,
                "Height": 0.5437348484992981
            }
        }
    },
    "UserMatches": [
        {
            "User": {
                "UserId": "demoUser1",
                "UserStatus": "ACTIVE"
            },
            "Similarity": 100.0
        },
        {
            "User": {
                "UserId": "demoUser2",
                "UserStatus": "ACTIVE"
            },
            "Similarity": 99.97946166992188
        }
    ],
    "FaceModelVersion": "6",
    "UnsearchedFaces": [
        {
            "FaceDetails": {
                "BoundingBox": {
                    "Width": 0.031677018851041794,
                    "Top": 0.5593535900115967,
                    "Left": 0.6102562546730042,
                    "Height": 0.0682177022099495
                }
            },
            "Reasons": [
                "FACE_NOT_LARGEST"
            ]
        },
        {
            "FaceDetails": {
                "BoundingBox": {
                    "Width": 0.03254449740052223,
```

```
                    "Top": 0.6080358028411865,
                    "Left": 0.516062319278717,
                    "Height": 0.06347997486591339
                }
            },
            "Reasons": [
                "FACE_NOT_LARGEST"
            ]
        }
    ]
}
```

Búsqueda de rostros en vídeos almacenados

Puede buscar una colección de rostros que coincida con rostros de personas detectados en un vídeo almacenado o un vídeo en streaming. En esta sección se explica la búsqueda de rostros en un vídeo almacenado. Para obtener información sobre la búsqueda de rostros en un vídeo en streaming, consulte Trabajar con eventos de vídeo en streaming.

Los rostros que busca se deben indexar primero en una colección utilizando IndexFaces. Para obtener más información, consulte Agregar rostros a una colección.

La búsqueda de rostros de Amazon Rekognition Video sigue el mismo flujo de trabajo asíncrono que otras operaciones de Amazon Rekognition Video que analizan vídeos almacenados en un bucket de Amazon S3. Para empezar a buscar rostros en un vídeo almacenado, llama StartFaceSearchy proporciona el identificador de la colección en la que deseas buscar. Amazon Rekognition Video publica el estado de finalización de una operación de análisis de vídeo en un tema de Amazon Simple Notification Service (Amazon SNS). Si el análisis de vídeo es correcto, llame a GetFaceSearch para obtener los resultados de la búsqueda. Para obtener más información sobre cómo iniciar el análisis de vídeo y obtener los resultados, consulte Cómo llamar a las operaciones de Amazon Rekognition Video.

El siguiente procedimiento muestra cómo buscar una colección de rostros que coincide con los rostros de las personas detectados en un vídeo. El procedimiento también muestra cómo obtener los datos de seguimiento de las personas que coinciden en el vídeo. El procedimiento amplía el código de Análisis de un vídeo almacenado en un bucket de Amazon S3 con Java o Python (SDK), que utiliza una cola de Amazon Simple Queue Service (Amazon SQS) para obtener el estado de realización de una solicitud de análisis de vídeo.

Para buscar rostros coincidentes en un vídeo (SDK)

1. Crear una colección.
2. Indexe un rostro en la colección.
3. Realice Análisis de un vídeo almacenado en un bucket de Amazon S3 con Java o Python (SDK).
4. Añada el código siguiente a la clase `VideoDetect` que ha creado en el paso 3.

   Java

```
   //Copyright 2018 Amazon.com, Inc. or its affiliates. All Rights Reserved.
   //PDX-License-Identifier: MIT-0 (For details, see https://github.com/awsdocs/
amazon-rekognition-developer-guide/blob/master/LICENSE-SAMPLECODE.)

       //Face collection search in video
 ======================================================================
       private static void StartFaceSearchCollection(String bucket, String
 video, String collection) throws Exception{

        NotificationChannel channel= new NotificationChannel()
                .withSNSTopicArn(snsTopicArn)
                .withRoleArn(roleArn);

        StartFaceSearchRequest req = new StartFaceSearchRequest()
                .withCollectionId(collection)
                .withVideo(new Video()
                        .withS3Object(new S3Object()
                                .withBucket(bucket)
                                .withName(video)))
                .withNotificationChannel(channel);



        StartFaceSearchResult startPersonCollectionSearchResult =
 rek.startFaceSearch(req);
        startJobId=startPersonCollectionSearchResult.getJobId();

    }

    //Face collection search in video
 ======================================================================
    private static void GetFaceSearchCollectionResults() throws Exception{
```

```
        GetFaceSearchResult faceSearchResult=null;
        int maxResults=10;
        String paginationToken=null;

        do {

            if (faceSearchResult !=null){
                paginationToken = faceSearchResult.getNextToken();
            }


            faceSearchResult  = rek.getFaceSearch(
                    new GetFaceSearchRequest()
                    .withJobId(startJobId)
                    .withMaxResults(maxResults)
                    .withNextToken(paginationToken)
                    .withSortBy(FaceSearchSortBy.TIMESTAMP)
                    );


            VideoMetadata videoMetaData=faceSearchResult.getVideoMetadata();

            System.out.println("Format: " + videoMetaData.getFormat());
            System.out.println("Codec: " + videoMetaData.getCodec());
            System.out.println("Duration: " +
videoMetaData.getDurationMillis());
            System.out.println("FrameRate: " + videoMetaData.getFrameRate());
            System.out.println();


            //Show search results
            List<PersonMatch> matches=
                    faceSearchResult.getPersons();

            for (PersonMatch match: matches) {
                long milliSeconds=match.getTimestamp();
                System.out.print("Timestamp: " + Long.toString(milliSeconds));
                System.out.println(" Person number: " +
match.getPerson().getIndex());
                List <FaceMatch> faceMatches = match.getFaceMatches();
                if (faceMatches != null) {
                    System.out.println("Matches in collection...");
                    for (FaceMatch faceMatch: faceMatches){
                        Face face=faceMatch.getFace();
```

```
                    System.out.println("Face Id: "+ face.getFaceId());
                    System.out.println("Similarity: " +
faceMatch.getSimilarity().toString());
                    System.out.println();
                }
            }
            System.out.println();
        }

        System.out.println();

     } while (faceSearchResult !=null && faceSearchResult.getNextToken() !=
null);

  }
```

En la función `main`, reemplace las líneas:

```
        StartLabelDetection(bucket, video);

        if (GetSQSMessageSuccess()==true)
         GetLabelDetectionResults();
```

por:

```
        String collection="collection";
        StartFaceSearchCollection(bucket, video, collection);

        if (GetSQSMessageSuccess()==true)
         GetFaceSearchCollectionResults();
```

Java V2

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.*;
import java.util.List;
```

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class VideoDetectFaces {
    private static String startJobId = "";

    public static void main(String[] args) {
        final String usage = """

                Usage:    <bucket> <video> <topicArn> <roleArn>

                Where:
                   bucket - The name of the bucket in which the video is located
 (for example, (for example, myBucket).\s
                   video - The name of video (for example, people.mp4).\s
                   topicArn - The ARN of the Amazon Simple Notification Service
 (Amazon SNS) topic.\s
                   roleArn - The ARN of the AWS Identity and Access Management
 (IAM) role to use.\s
                """;

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

        String bucket = args[0];
        String video = args[1];
        String topicArn = args[2];
        String roleArn = args[3];

        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        NotificationChannel channel = NotificationChannel.builder()
                .snsTopicArn(topicArn)
```

```
                .roleArn(roleArn)
                .build();

        startFaceDetection(rekClient, channel, bucket, video);
        getFaceResults(rekClient);
        System.out.println("This example is done!");
        rekClient.close();
    }

    public static void startFaceDetection(RekognitionClient rekClient,
            NotificationChannel channel,
            String bucket,
            String video) {
        try {
            S3Object s3Obj = S3Object.builder()
                    .bucket(bucket)
                    .name(video)
                    .build();

            Video vidOb = Video.builder()
                    .s3Object(s3Obj)
                    .build();

            StartFaceDetectionRequest faceDetectionRequest =
StartFaceDetectionRequest.builder()
                    .jobTag("Faces")
                    .faceAttributes(FaceAttributes.ALL)
                    .notificationChannel(channel)
                    .video(vidOb)
                    .build();

            StartFaceDetectionResponse startLabelDetectionResult =
rekClient.startFaceDetection(faceDetectionRequest);
            startJobId = startLabelDetectionResult.jobId();

        } catch (RekognitionException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void getFaceResults(RekognitionClient rekClient) {
        try {
            String paginationToken = null;
```

```
            GetFaceDetectionResponse faceDetectionResponse = null;
            boolean finished = false;
            String status;
            int yy = 0;

            do {
                if (faceDetectionResponse != null)
                    paginationToken = faceDetectionResponse.nextToken();

                GetFaceDetectionRequest recognitionRequest =
GetFaceDetectionRequest.builder()
                        .jobId(startJobId)
                        .nextToken(paginationToken)
                        .maxResults(10)
                        .build();

                // Wait until the job succeeds.
                while (!finished) {

                    faceDetectionResponse =
rekClient.getFaceDetection(recognitionRequest);
                    status = faceDetectionResponse.jobStatusAsString();

                    if (status.compareTo("SUCCEEDED") == 0)
                        finished = true;
                    else {
                        System.out.println(yy + " status is: " + status);
                        Thread.sleep(1000);
                    }
                    yy++;
                }

                finished = false;

                // Proceed when the job is done - otherwise VideoMetadata is
null.
                VideoMetadata videoMetaData =
faceDetectionResponse.videoMetadata();
                System.out.println("Format: " + videoMetaData.format());
                System.out.println("Codec: " + videoMetaData.codec());
                System.out.println("Duration: " +
videoMetaData.durationMillis());
                System.out.println("FrameRate: " + videoMetaData.frameRate());
                System.out.println("Job");
```

```
                // Show face information.
                List<FaceDetection> faces = faceDetectionResponse.faces();
                for (FaceDetection face : faces) {
                    String age = face.face().ageRange().toString();
                    String smile = face.face().smile().toString();
                    System.out.println("The detected face is estimated to be"
                            + age + " years old.");
                    System.out.println("There is a smile : " + smile);
                }

            } while (faceDetectionResponse != null &&
 faceDetectionResponse.nextToken() != null);

        } catch (RekognitionException | InterruptedException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

Python

```
#Copyright 2018 Amazon.com, Inc. or its affiliates. All Rights Reserved.
#PDX-License-Identifier: MIT-0 (For details, see https://github.com/awsdocs/
amazon-rekognition-developer-guide/blob/master/LICENSE-SAMPLECODE.)

    # ============== Face Search ===============
    def StartFaceSearchCollection(self,collection):
        response = self.rek.start_face_search(Video={'S3Object':
{'Bucket':self.bucket,'Name':self.video}},
            CollectionId=collection,
            NotificationChannel={'RoleArn':self.roleArn,
 'SNSTopicArn':self.snsTopicArn})

        self.startJobId=response['JobId']

        print('Start Job Id: ' + self.startJobId)


    def GetFaceSearchCollectionResults(self):
        maxResults = 10
        paginationToken = ''
```

```
        finished = False

        while finished == False:
            response = self.rek.get_face_search(JobId=self.startJobId,
                                        MaxResults=maxResults,
                                        NextToken=paginationToken)

            print(response['VideoMetadata']['Codec'])
            print(str(response['VideoMetadata']['DurationMillis']))
            print(response['VideoMetadata']['Format'])
            print(response['VideoMetadata']['FrameRate'])

            for personMatch in response['Persons']:
                print('Person Index: ' + str(personMatch['Person']['Index']))
                print('Timestamp: ' + str(personMatch['Timestamp']))

                if ('FaceMatches' in personMatch):
                    for faceMatch in personMatch['FaceMatches']:
                        print('Face ID: ' + faceMatch['Face']['FaceId'])
                        print('Similarity: ' + str(faceMatch['Similarity']))
                print()
            if 'NextToken' in response:
                paginationToken = response['NextToken']
            else:
                finished = True
            print()
```

En la función `main`, reemplace las líneas:

```
    analyzer.StartLabelDetection()
    if analyzer.GetSQSMessageSuccess()==True:
        analyzer.GetLabelDetectionResults()
```

por:

```
    collection='tests'
    analyzer.StartFaceSearchCollection(collection)

    if analyzer.GetSQSMessageSuccess()==True:
        analyzer.GetFaceSearchCollectionResults()
```

Si ya ha ejecutado un ejemplo de vídeo distinto de Análisis de un vídeo almacenado en un bucket de Amazon S3 con Java o Python (SDK), el código que se va a reemplazar podría ser diferente.

5. Cambie el valor de `collection` por el nombre de la colección que ha creado en el paso 1.
6. Ejecute el código. Se muestra una lista de personas del vídeo cuyos rostros coinciden con los de la colección de entrada. También se muestran los datos de seguimiento de cada persona que coincida.

GetFaceSearch respuesta de operación

A continuación se muestra un ejemplo de respuesta JSON de `GetFaceSearch`.

La respuesta incluye una matriz de personas (`Persons`), detectadas en el vídeo cuyos rostro(s) coinciden con un rostro de la colección de entrada. Un elemento de matriz, PersonMatch, existe para cada vez que la persona coincide en el vídeo. Cada `PersonMatch` incluye una gama de coincidencias de rostros a partir de la colección de entrada, FaceMatch, la información sobre la persona que coincide, PersonDetail, y el momento en que se encontró una coincidencia de la persona en el vídeo.


```
{
    "JobStatus": "SUCCEEDED",
    "NextToken": "IJdbzkZfvBRqj8GPV82BPiZKkLOGCqDIsNZG/gQsEE5faTVK9JHOz/
xxxxxxxxxxxxxx",
    "Persons": [
        {
            "FaceMatches": [
                {
                    "Face": {
                        "BoundingBox": {
                            "Height": 0.527472972869873,
                            "Left": 0.33530598878860474,
                            "Top": 0.2161169946193695,
                            "Width": 0.35503000020980835
                        },
                        "Confidence": 99.90239715576172,
                        "ExternalImageId": "image.PNG",
                        "FaceId": "a2f2e224-bfaa-456c-b360-7c00241e5e2d",
                        "ImageId": "eb57ed44-8d8d-5ec5-90b8-6d190daff4c3"
```

```
                },
                "Similarity": 98.40909576416016
            }
        ],
        "Person": {
            "BoundingBox": {
                "Height": 0.8694444298744202,
                "Left": 0.2473958283662796,
                "Top": 0.10092592239379883,
                "Width": 0.49427083134651184
            },
            "Face": {
                "BoundingBox": {
                    "Height": 0.23000000417232513,
                    "Left": 0.42500001192092896,
                    "Top": 0.16333332657814026,
                    "Width": 0.12937499582767487
                },
                "Confidence": 99.97504425048828,
                "Landmarks": [
                    {
                        "Type": "eyeLeft",
                        "X": 0.46415066719055176,
                        "Y": 0.2572723925113678
                    },
                    {
                        "Type": "eyeRight",
                        "X": 0.5068183541297913,
                        "Y": 0.23705792427062988
                    },
                    {
                        "Type": "nose",
                        "X": 0.49765899777412415,
                        "Y": 0.28383663296699524
                    },
                    {
                        "Type": "mouthLeft",
                        "X": 0.487221896648407,
                        "Y": 0.3452930748462677
                    },
                    {
                        "Type": "mouthRight",
                        "X": 0.5142884850502014,
                        "Y": 0.33167609572410583
```

```
                    }
                ],
                "Pose": {
                    "Pitch": 15.966927528381348,
                    "Roll": -15.547388076782227,
                    "Yaw": 11.34195613861084
                },
                "Quality": {
                    "Brightness": 44.80223083496094,
                    "Sharpness": 99.95819854736328
                }
            },
            "Index": 0
        },
        "Timestamp": 0
    },
    {
        "Person": {
            "BoundingBox": {
                "Height": 0.2177777737379074,
                "Left": 0.7593749761581421,
                "Top": 0.13333334028720856,
                "Width": 0.12250000238418579
            },
            "Face": {
                "BoundingBox": {
                    "Height": 0.2177777737379074,
                    "Left": 0.7593749761581421,
                    "Top": 0.13333334028720856,
                    "Width": 0.12250000238418579
                },
                "Confidence": 99.63436889648438,
                "Landmarks": [
                    {
                        "Type": "eyeLeft",
                        "X": 0.8005779385566711,
                        "Y": 0.20915353298187256
                    },
                    {
                        "Type": "eyeRight",
                        "X": 0.8391435146331787,
                        "Y": 0.21049551665782928
                    },
                    {
```

```
                        "Type": "nose",
                        "X": 0.8191410899162292,
                        "Y": 0.2523227035999298
                    },
                    {
                        "Type": "mouthLeft",
                        "X": 0.8093273043632507,
                        "Y": 0.29053622484207153
                    },
                    {
                        "Type": "mouthRight",
                        "X": 0.8366993069648743,
                        "Y": 0.29101791977882385
                    }
                ],
                "Pose": {
                    "Pitch": 3.165884017944336,
                    "Roll": 1.4182015657424927,
                    "Yaw": -11.151537895202637
                },
                "Quality": {
                    "Brightness": 28.910892486572266,
                    "Sharpness": 97.61507415771484
                }
            },
            "Index": 1
        },
        "Timestamp": 0
    },
    {
        "Person": {
            "BoundingBox": {
                "Height": 0.8388888835906982,
                "Left": 0,
                "Top": 0.15833333134651184,
                "Width": 0.2369791716337204
            },
            "Face": {
                "BoundingBox": {
                    "Height": 0.20000000298023224,
                    "Left": 0.029999999329447746,
                    "Top": 0.2199999988079071,
                    "Width": 0.11249999701976776
                },
```

```
                    "Confidence": 99.85971069335938,
                    "Landmarks": [
                        {
                            "Type": "eyeLeft",
                            "X": 0.06842322647571564,
                            "Y": 0.3010137975215912
                        },
                        {
                            "Type": "eyeRight",
                            "X": 0.10543643683195114,
                            "Y": 0.29697132110595703
                        },
                        {
                            "Type": "nose",
                            "X": 0.09569807350635529,
                            "Y": 0.33701086044311523
                        },
                        {
                            "Type": "mouthLeft",
                            "X": 0.0732642263174057,
                            "Y": 0.3757539987564087
                        },
                        {
                            "Type": "mouthRight",
                            "X": 0.10589495301246643,
                            "Y": 0.3722417950630188
                        }
                    ],
                    "Pose": {
                        "Pitch": -0.5589138865470886,
                        "Roll": -5.1093974113464355,
                        "Yaw": 18.69594955444336
                    },
                    "Quality": {
                        "Brightness": 43.052337646484375,
                        "Sharpness": 99.68138885498047
                    }
                },
                "Index": 2
            },
            "Timestamp": 0
        }......

    ],
```

```
    "VideoMetadata": {
        "Codec": "h264",
        "DurationMillis": 67301,
        "Format": "QuickTime / MOV",
        "FrameHeight": 1080,
        "FrameRate": 29.970029830932617,
        "FrameWidth": 1920
    }
}
```

Búsqueda de rostros en una colección en streaming de vídeo

Puede usar Amazon Rekognition Video para detectar y reconocer rostros de una colección en vídeo en streaming. Con Amazon Rekognition Video, puede crear un procesador de transmisión CreateStreamProcessor() para iniciar y administrar el análisis de la transmisión de vídeo.

Para detectar un rostro conocido en una transmisión de vídeo (búsqueda de rostros), Amazon Rekognition Video utiliza Amazon Kinesis Video Streams para recibir y procesar una transmisión de vídeo. Los resultados del análisis se envían de Amazon Rekognition Video a un flujo de datos de Kinesis y, a continuación, los lee la aplicación cliente.

Para utilizar Amazon Rekognition Video con vídeo en streaming, la aplicación tiene que implementar lo siguiente:

- Una transmisión de vídeo de Kinesis para enviar vídeo en streaming a Amazon Rekognition Video. Para obtener más información, consulte la Guía para desarrolladores de Amazon Kinesis Vídeo Streams.
- Un procesador de streaming de Amazon Rekognition Video para administrar el análisis del vídeo en streaming. Para obtener más información, consulte Descripción general de las operaciones del procesador de transmisión de Amazon Rekognition Video.
- Un consumidor de flujos de datos de Kinesis que lea los resultados de los análisis que Amazon Rekognition Video envía al flujo de datos de Kinesis. Para obtener más información, consulte Kinesis Data Streams Consumers.

Esta sección contiene información sobre cómo crear una aplicación que cree la transmisión de vídeo de Kinesis y otros recursos necesarios, transmita vídeo a Amazon Rekognition Video y reciba los resultados del análisis.

Temas

- Configuración de los recursos de Amazon Rekognition Video y Amazon Kinesis
- Búsqueda de rostros en una transmisión de vídeo
- Transmisión mediante un GStreamer complemento
- Solución de problemas de vídeo en streaming

Configuración de los recursos de Amazon Rekognition Video y Amazon Kinesis

Los siguientes procedimientos describen los pasos que debe seguir para aprovisionar la transmisión de vídeo de Kinesis y otros recursos que se utilizan para reconocer rostros en una transmisión de vídeo.

Requisitos previos

Para ejecutar este procedimiento, debe tener el instalado. AWS SDK para Java Para obtener más información, consulte Introducción a Amazon Rekognition. El Cuenta de AWS que utilices debe tener permisos de acceso a la API Amazon Rekognition. Para obtener más información, consulte Acciones definidas por Amazon Rekognition en la Guía del usuario de IAM.

Para reconocer rostros en una transmisión de vídeo (AWS SDK)

1. Si no lo ha hecho aún, cree un rol de servicio de IAM para otorgar a Amazon Rekognition Video acceso a sus transmisiones de vídeo de Kinesis y sus flujos de datos de Kinesis. Anote el ARN. Para obtener más información, consulte Dar acceso a las transmisiones mediante AmazonRekognitionServiceRole.
2. Cree una colección y anote el identificador de la colección que haya utilizado.
3. Indexe los rostros que desee buscar en la colección que ha creado en el paso 2.
4. Cree una transmisión de vídeo de Kinesis y anote el Nombre de recurso de Amazon (ARN) de la transmisión.
5. Cree de un flujo de datos de Kinesis. Añada el nombre de la transmisión AmazonRekognitiony anote el ARN de la transmisión.

A continuación, puede crear el procesador de transmisión por búsqueda de rostros e iniciar el procesador de transmisión que haya elegido.

> **Note**
>
> Debe iniciar el procesador de transmisión solo después de comprobar que puede introducir contenido multimedia en la transmisión de vídeo de Kinesis.

Transmisión de vídeo a Amazon Rekognition Video

Para transmitir vídeo a Amazon Rekognition Video, utilice el SDK de Amazon Kinesis Video Streams para crear y utilizar una transmisión de vídeo de Kinesis. La operación `PutMedia` escribe fragmentos de datos de vídeo de Kinesis en una transmisión de vídeo que Amazon Rekognition Video consume. Cada fragmento de datos de vídeo suele tener una longitud de 2 a 10 segundos y contiene una secuencia de fotogramas de vídeo autónoma. Amazon Rekognition Video admite vídeos cifrados en H.264, que pueden tener tres tipos de fotogramas (I, B y P). Para obtener más información, consulte Inter Frame. El primer fotograma del fragmento debe ser un I-frame. Un I-frame se puede decodificar de forma independientes de cualquier otro fotograma.

A medida que los datos de vídeo llegan a transmisión de vídeo de Kinesis, Kinesis Video Streams asigna un número único al fragmento. Para ver un ejemplo, consulta el ejemplo de APIPutMedia .

- Si está transmitiendo desde una fuente codificada en Matroska (MKV), utilice la PutMediaoperación para transmitir el vídeo de origen a la transmisión de vídeo de Kinesis que ha creado. Para obtener más información, consulte el ejemplo de API. PutMedia
- Si está transmitiendo desde la cámara de un dispositivo, consulte Transmisión mediante un GStreamer complemento.

Otorgar a Amazon Rekognition Video acceso a sus recursos

Utiliza un rol de servicio AWS Identity and Access Management (IAM) para dar a Amazon Rekognition Video acceso de lectura a las transmisiones de vídeo de Kinesis. Si utiliza un procesador de secuencias de búsqueda facial, utilizará un rol de servicio de IAM para conceder a Amazon Rekognition Video acceso de escritura a los flujos de datos de Kinesis. Si utiliza un procesador de flujos de supervisión de seguridad, utiliza los roles de IAM para permitir que Amazon Rekognition Video acceda a su bucket de Amazon S3 y a un tema de Amazon SNS.

Permitir el acceso a los procesadores de streaming mediante búsqueda facial

Puede crear una política de permisos que permita a Amazon Rekognition Video acceder a las transmisiones individuales de vídeo de Kinesis y a los flujos de datos de Kinesis.

Para dar acceso a Amazon Rekognition Video a un procesador de secuencias de búsqueda facial

1. Cree una nueva política de permisos con el editor de políticas de JSON de IAM y utilice la política siguiente. Sustituya `video-arn` por el ARN de la transmisión de vídeo de Kinesis deseado. Si utiliza un procesador de secuencias de búsqueda facial, sustituya `data-arn` por el ARN del flujo de datos de Kinesis deseado.
2. Cree un rol de servicio de IAM o actualice un rol de servicio de IAM existente. Utilice la siguiente información para crear el rol de servicio de IAM:
    1. Elija Rekognition para el nombre del servicio.
    2. Elija Rekognition para el caso de uso del rol de servicio.
    3. Adjunte la política de permisos que ha creado en el paso 1.
3. Anote el ARN del rol de servicio. Lo necesita para comenzar las operaciones de análisis de vídeo.

Dar acceso a las transmisiones mediante AmazonRekognitionServiceRole

Como opción alternativa para configurar el acceso a las transmisiones de vídeo y flujos de datos de Kinesis, puede utilizar la política de permisos `AmazonRekognitionServiceRole`. IAM proporciona el caso de uso del rol de servicio Rekognition que, cuando se usa con la política de permisos `AmazonRekognitionServiceRole`, puede escribir en varios flujo de datos de Kinesis y leer desde todas transmisiones de vídeo de Kinesis. Para conceder a Amazon Rekognition Video acceso de escritura a varias transmisiones de datos de Kinesis, puede anteponer los nombres de las transmisiones de datos de Kinesis con, por ejemplo,. AmazonRekognition`AmazonRekognitionMyDataStreamName`

Para conceder a Amazon Rekognition Video acceso a su transmisión de vídeo de Kinesis y a los flujos de datos de Kinesis

1. Cree un rol de servicio de IAM. Utilice la siguiente información para crear el rol de servicio de IAM:
    1. Elija Rekognition para el nombre del servicio.

2. Elija Rekognition para el caso de uso del rol de servicio.
3. Elija la política de AmazonRekognitionServiceRolepermisos, que otorga a Amazon Rekognition Video acceso de escritura a las transmisiones de datos de Kinesis que tienen el prefijo AmazonRekognitiony acceso de lectura a todas sus transmisiones de vídeo de Kinesis.

2. Para garantizar su Cuenta de AWS seguridad, limite el alcance del acceso de Rekognition únicamente a los recursos que utilice. Para ello, puede adjuntar una política de confianza a su rol de servicio de IAM. Para obtener información sobre cómo hacerlo, consulte Prevención de la sustitución confusa entre servicios.
3. Anote el nombre de recurso de Amazon (ARN) del rol de servicio. Lo necesita para comenzar las operaciones de análisis de vídeo.

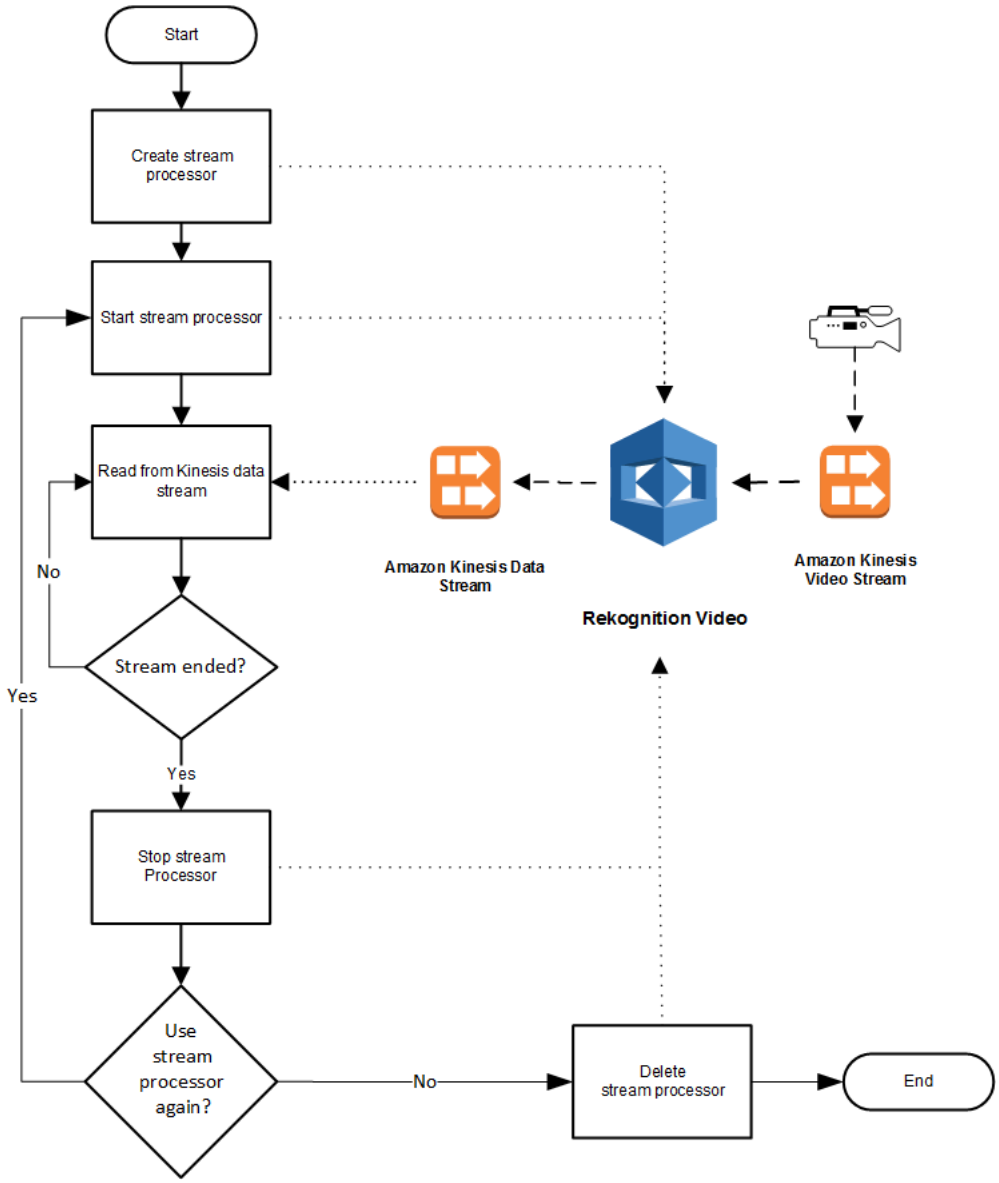
Búsqueda de rostros en una transmisión de vídeo

Amazon Rekognition Video puede buscar rostros en una colección que coincidan con rostros detectados en un vídeo en streaming. Para obtener más información sobre las colecciones, consulte Búsqueda de rostros en una colección.

Temas

- Creación del procesador de flujo de búsqueda de rostros de Amazon Rekognition Video
- Iniciar el procesador de flujo de búsqueda de rostros de Amazon Rekognition Video
- Uso de procesadores de flujo para la búsqueda de rostros (ejemplo de Java V2)
- Uso de procesadores de flujo para la búsqueda de rostros (ejemplo de Java V1)
- Lectura de los resultados del análisis de vídeo en streaming
- Visualización local de los resultados de Rekognition con Kinesis Video Streams
- Descripción del registro de fotogramas de JSON para reconocimiento facial de Kinesis

El siguiente diagrama muestra cómo Amazon Rekognition Video detecta y reconoce rostros en vídeo en streaming.



Creación del procesador de flujo de búsqueda de rostros de Amazon Rekognition Video

Antes de poder analizar un vídeo en streaming, debe crear un procesador de streaming Amazon Rekognition Video (). CreateStreamProcessor El procesador de transmisión contiene información sobre el flujo de datos de Kinesis y la transmisión de vídeo de Kinesis. También contiene el identificador de la colección que contiene los rostros que desea reconocer en el vídeo de streaming de entrada. Además especifica un nombre para el procesador de streaming. A continuación, se muestra un ejemplo de JSON para la solicitud `CreateStreamProcessor`.

```
{
       "Name": "streamProcessorForCam",
```

```
	"Input": {
		"KinesisVideoStream": {
			"Arn": "arn:aws:kinesisvideo:us-east-1:nnnnnnnnnnnn:stream/
inputVideo"
		}
	},
	"Output": {
		"KinesisDataStream": {
			"Arn": "arn:aws:kinesis:us-east-1:nnnnnnnnnnnn:stream/outputData"
		}
	},
	"RoleArn": "arn:aws:iam::nnnnnnnnnnn:role/roleWithKinesisPermission",
	"Settings": {
		"FaceSearch": {
			"CollectionId": "collection-with-100-faces",
			"FaceMatchThreshold": 85.5
		}
	}
}
```

A continuación se muestra un ejemplo de respuesta de `CreateStreamProcessor`.

```
{
	“StreamProcessorArn”: “arn:aws:rekognition:us-
east-1:nnnnnnnnnnnn:streamprocessor/streamProcessorForCam”
}
```

Iniciar el procesador de flujo de búsqueda de rostros de Amazon Rekognition Video

Se comienza a analizar el vídeo en streaming llamando a StartStreamProcessor con el nombre del procesador de streaming que especificó en `CreateStreamProcessor`. A continuación, se muestra un ejemplo de JSON para la solicitud `StartStreamProcessor`.

```
{
	"Name": "streamProcessorForCam"
}
```

Si el procesador de streaming comienza correctamente, se devuelve una respuesta HTTP 200, junto con un cuerpo JSON vacío.

Uso de procesadores de flujo para la búsqueda de rostros (ejemplo de Java V2)

El siguiente código de ejemplo muestra cómo llamar a varias operaciones del procesador de flujo, como CreateStreamProcessory StartStreamProcessormediante el uso de la versión 2 del AWS SDK for Java.

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.CreateStreamProcessorRequest;
import software.amazon.awssdk.services.rekognition.model.CreateStreamProcessorResponse;
import software.amazon.awssdk.services.rekognition.model.FaceSearchSettings;
import software.amazon.awssdk.services.rekognition.model.KinesisDataStream;
import software.amazon.awssdk.services.rekognition.model.KinesisVideoStream;
import software.amazon.awssdk.services.rekognition.model.ListStreamProcessorsRequest;
import software.amazon.awssdk.services.rekognition.model.ListStreamProcessorsResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import software.amazon.awssdk.services.rekognition.model.StreamProcessor;
import software.amazon.awssdk.services.rekognition.model.StreamProcessorInput;
import software.amazon.awssdk.services.rekognition.model.StreamProcessorSettings;
import software.amazon.awssdk.services.rekognition.model.StreamProcessorOutput;
import software.amazon.awssdk.services.rekognition.model.StartStreamProcessorRequest;
import
 software.amazon.awssdk.services.rekognition.model.DescribeStreamProcessorRequest;
import
 software.amazon.awssdk.services.rekognition.model.DescribeStreamProcessorResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateStreamProcessor {
    public static void main(String[] args) {
        final String usage = """
```

```
                Usage:    <role> <kinInputStream> <kinOutputStream> <collectionName>
<StreamProcessorName>

                Where:
                   role - The ARN of the AWS Identity and Access Management (IAM) role
to use. \s
                   kinInputStream - The ARN of the Kinesis video stream.\s
                   kinOutputStream - The ARN of the Kinesis data stream.\s
                   collectionName - The name of the collection to use that contains
content. \s
                   StreamProcessorName - The name of the Stream Processor. \s
                """;

       if (args.length != 5) {
           System.out.println(usage);
           System.exit(1);
       }

       String role = args[0];
       String kinInputStream = args[1];
       String kinOutputStream = args[2];
       String collectionName = args[3];
       String streamProcessorName = args[4];

       Region region = Region.US_EAST_1;
       RekognitionClient rekClient = RekognitionClient.builder()
               .region(region)
               .build();

       processCollection(rekClient, streamProcessorName, kinInputStream,
kinOutputStream, collectionName,
               role);
       startSpecificStreamProcessor(rekClient, streamProcessorName);
       listStreamProcessors(rekClient);
       describeStreamProcessor(rekClient, streamProcessorName);
       deleteSpecificStreamProcessor(rekClient, streamProcessorName);
   }

   public static void listStreamProcessors(RekognitionClient rekClient) {
       ListStreamProcessorsRequest request = ListStreamProcessorsRequest.builder()
               .maxResults(15)
               .build();
```

```
        ListStreamProcessorsResponse listStreamProcessorsResult =
rekClient.listStreamProcessors(request);
        for (StreamProcessor streamProcessor :
listStreamProcessorsResult.streamProcessors()) {
            System.out.println("StreamProcessor name - " + streamProcessor.name());
            System.out.println("Status - " + streamProcessor.status());
        }
    }

    private static void describeStreamProcessor(RekognitionClient rekClient, String
StreamProcessorName) {
        DescribeStreamProcessorRequest streamProcessorRequest =
DescribeStreamProcessorRequest.builder()
                .name(StreamProcessorName)
                .build();

        DescribeStreamProcessorResponse describeStreamProcessorResult = rekClient
                .describeStreamProcessor(streamProcessorRequest);
        System.out.println("Arn - " +
describeStreamProcessorResult.streamProcessorArn());
        System.out.println("Input kinesisVideo stream - "
                + describeStreamProcessorResult.input().kinesisVideoStream().arn());
        System.out.println("Output kinesisData stream - "
                + describeStreamProcessorResult.output().kinesisDataStream().arn());
        System.out.println("RoleArn - " + describeStreamProcessorResult.roleArn());
        System.out.println(
                "CollectionId - "
                        +
describeStreamProcessorResult.settings().faceSearch().collectionId());
        System.out.println("Status - " + describeStreamProcessorResult.status());
        System.out.println("Status message - " +
describeStreamProcessorResult.statusMessage());
        System.out.println("Creation timestamp - " +
describeStreamProcessorResult.creationTimestamp());
        System.out.println("Last update timestamp - " +
describeStreamProcessorResult.lastUpdateTimestamp());
    }

    private static void startSpecificStreamProcessor(RekognitionClient rekClient,
String StreamProcessorName) {
        try {
            StartStreamProcessorRequest streamProcessorRequest =
StartStreamProcessorRequest.builder()
                    .name(StreamProcessorName)
```

```
                    .build();

            rekClient.startStreamProcessor(streamProcessorRequest);
            System.out.println("Stream Processor " + StreamProcessorName + "
started.");

        } catch (RekognitionException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }

    private static void processCollection(RekognitionClient rekClient, String
StreamProcessorName,
                                          String kinInputStream, String
kinOutputStream, String collectionName, String role) {
        try {
            KinesisVideoStream videoStream = KinesisVideoStream.builder()
                    .arn(kinInputStream)
                    .build();

            KinesisDataStream dataStream = KinesisDataStream.builder()
                    .arn(kinOutputStream)
                    .build();

            StreamProcessorOutput processorOutput = StreamProcessorOutput.builder()
                    .kinesisDataStream(dataStream)
                    .build();

            StreamProcessorInput processorInput = StreamProcessorInput.builder()
                    .kinesisVideoStream(videoStream)
                    .build();

            FaceSearchSettings searchSettings = FaceSearchSettings.builder()
                    .faceMatchThreshold(75f)
                    .collectionId(collectionName)
                    .build();

            StreamProcessorSettings processorSettings =
StreamProcessorSettings.builder()
                    .faceSearch(searchSettings)
                    .build();
```

```
            CreateStreamProcessorRequest processorRequest =
 CreateStreamProcessorRequest.builder()
                    .name(StreamProcessorName)
                    .input(processorInput)
                    .output(processorOutput)
                    .roleArn(role)
                    .settings(processorSettings)
                    .build();

            CreateStreamProcessorResponse response =
 rekClient.createStreamProcessor(processorRequest);
            System.out.println("The ARN for the newly create stream processor is "
                    + response.streamProcessorArn());

        } catch (RekognitionException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }

    private static void deleteSpecificStreamProcessor(RekognitionClient rekClient,
 String StreamProcessorName) {
        rekClient.stopStreamProcessor(a -> a.name(StreamProcessorName));
        rekClient.deleteStreamProcessor(a -> a.name(StreamProcessorName));
        System.out.println("Stream Processor " + StreamProcessorName + " deleted.");
    }
}
```

Uso de procesadores de flujo para la búsqueda de rostros (ejemplo de Java V1)

El siguiente código de ejemplo muestra cómo llamar a varias operaciones del procesador de flujos, como CreateStreamProcessory StartStreamProcessormediante Java V1. El ejemplo incluye una clase de administrador de procesadores de flujo (StreamManager) que proporciona métodos para llamar a las operaciones del procesador de flujo. La clase Starter (Starter) crea un StreamManager objeto y llama a varias operaciones.

Para configurar el ejemplo:

1. Establezca los valores de los campos de miembro de la clase Starter a los valores que desee.
2. En la función de clase Starter `main`, quite el comentario de la llamada de función que desee.

Clase Starter

```
//Copyright 2018 Amazon.com, Inc. or its affiliates. All Rights Reserved.
//PDX-License-Identifier: MIT-0 (For details, see https://github.com/awsdocs/amazon-
rekognition-developer-guide/blob/master/LICENSE-SAMPLECODE.)

// Starter class. Use to create a StreamManager class
// and call stream processor operations.
package com.amazonaws.samples;
import com.amazonaws.samples.*;

public class Starter {

 public static void main(String[] args) {


     String streamProcessorName="Stream Processor Name";
     String kinesisVideoStreamArn="Kinesis Video Stream Arn";
     String kinesisDataStreamArn="Kinesis Data Stream Arn";
     String roleArn="Role Arn";
     String collectionId="Collection ID";
     Float matchThreshold=50F;

  try {
   StreamManager sm= new StreamManager(streamProcessorName,
     kinesisVideoStreamArn,
     kinesisDataStreamArn,
     roleArn,
     collectionId,
     matchThreshold);
   //sm.createStreamProcessor();
   //sm.startStreamProcessor();
   //sm.deleteStreamProcessor();
   //sm.deleteStreamProcessor();
   //sm.stopStreamProcessor();
   //sm.listStreamProcessors();
   //sm.describeStreamProcessor();
  }
  catch(Exception e){
   System.out.println(e.getMessage());
  }
 }
}
```

StreamManager clase

```
//Copyright 2018 Amazon.com, Inc. or its affiliates. All Rights Reserved.
//PDX-License-Identifier: MIT-0 (For details, see https://github.com/awsdocs/amazon-
rekognition-developer-guide/blob/master/LICENSE-SAMPLECODE.)

// Stream manager class. Provides methods for calling
// Stream Processor operations.
package com.amazonaws.samples;

import com.amazonaws.services.rekognition.AmazonRekognition;
import com.amazonaws.services.rekognition.AmazonRekognitionClientBuilder;
import com.amazonaws.services.rekognition.model.CreateStreamProcessorRequest;
import com.amazonaws.services.rekognition.model.CreateStreamProcessorResult;
import com.amazonaws.services.rekognition.model.DeleteStreamProcessorRequest;
import com.amazonaws.services.rekognition.model.DeleteStreamProcessorResult;
import com.amazonaws.services.rekognition.model.DescribeStreamProcessorRequest;
import com.amazonaws.services.rekognition.model.DescribeStreamProcessorResult;
import com.amazonaws.services.rekognition.model.FaceSearchSettings;
import com.amazonaws.services.rekognition.model.KinesisDataStream;
import com.amazonaws.services.rekognition.model.KinesisVideoStream;
import com.amazonaws.services.rekognition.model.ListStreamProcessorsRequest;
import com.amazonaws.services.rekognition.model.ListStreamProcessorsResult;
import com.amazonaws.services.rekognition.model.StartStreamProcessorRequest;
import com.amazonaws.services.rekognition.model.StartStreamProcessorResult;
import com.amazonaws.services.rekognition.model.StopStreamProcessorRequest;
import com.amazonaws.services.rekognition.model.StopStreamProcessorResult;
import com.amazonaws.services.rekognition.model.StreamProcessor;
import com.amazonaws.services.rekognition.model.StreamProcessorInput;
import com.amazonaws.services.rekognition.model.StreamProcessorOutput;
import com.amazonaws.services.rekognition.model.StreamProcessorSettings;

public class StreamManager {

    private String streamProcessorName;
    private String kinesisVideoStreamArn;
    private String kinesisDataStreamArn;
    private String roleArn;
    private String collectionId;
    private float matchThreshold;
```

```
    private AmazonRekognition rekognitionClient;


    public StreamManager(String spName,
      String kvStreamArn,
      String kdStreamArn,
      String iamRoleArn,
      String collId,
      Float threshold){
     streamProcessorName=spName;
     kinesisVideoStreamArn=kvStreamArn;
     kinesisDataStreamArn=kdStreamArn;
     roleArn=iamRoleArn;
     collectionId=collId;
     matchThreshold=threshold;
     rekognitionClient=AmazonRekognitionClientBuilder.defaultClient();

    }

    public void createStreamProcessor() {
     //Setup input parameters
        KinesisVideoStream kinesisVideoStream = new
KinesisVideoStream().withArn(kinesisVideoStreamArn);
        StreamProcessorInput streamProcessorInput =
                new StreamProcessorInput().withKinesisVideoStream(kinesisVideoStream);
        KinesisDataStream kinesisDataStream = new
KinesisDataStream().withArn(kinesisDataStreamArn);
        StreamProcessorOutput streamProcessorOutput =
                new StreamProcessorOutput().withKinesisDataStream(kinesisDataStream);
        FaceSearchSettings faceSearchSettings =
                new
FaceSearchSettings().withCollectionId(collectionId).withFaceMatchThreshold(matchThreshold);
        StreamProcessorSettings streamProcessorSettings =
                new StreamProcessorSettings().withFaceSearch(faceSearchSettings);

        //Create the stream processor
        CreateStreamProcessorResult createStreamProcessorResult =
rekognitionClient.createStreamProcessor(
                new
CreateStreamProcessorRequest().withInput(streamProcessorInput).withOutput(streamProcessorOutpu

.withSettings(streamProcessorSettings).withRoleArn(roleArn).withName(streamProcessorName));

        //Display result
```

```
        System.out.println("Stream Processor " + streamProcessorName + " created.");
        System.out.println("StreamProcessorArn - " +
createStreamProcessorResult.getStreamProcessorArn());
    }

    public void startStreamProcessor() {
        StartStreamProcessorResult startStreamProcessorResult =
                rekognitionClient.startStreamProcessor(new
StartStreamProcessorRequest().withName(streamProcessorName));
        System.out.println("Stream Processor " + streamProcessorName + " started.");
    }

    public void stopStreamProcessor() {
        StopStreamProcessorResult stopStreamProcessorResult =
                rekognitionClient.stopStreamProcessor(new
StopStreamProcessorRequest().withName(streamProcessorName));
        System.out.println("Stream Processor " + streamProcessorName + " stopped.");
    }

    public void deleteStreamProcessor() {
        DeleteStreamProcessorResult deleteStreamProcessorResult = rekognitionClient
                .deleteStreamProcessor(new
DeleteStreamProcessorRequest().withName(streamProcessorName));
        System.out.println("Stream Processor " + streamProcessorName + " deleted.");
    }

    public void describeStreamProcessor() {
        DescribeStreamProcessorResult describeStreamProcessorResult = rekognitionClient
                .describeStreamProcessor(new
DescribeStreamProcessorRequest().withName(streamProcessorName));

        //Display various stream processor attributes.
        System.out.println("Arn - " +
describeStreamProcessorResult.getStreamProcessorArn());
        System.out.println("Input kinesisVideo stream - "
                +
describeStreamProcessorResult.getInput().getKinesisVideoStream().getArn());
        System.out.println("Output kinesisData stream - "
                +
describeStreamProcessorResult.getOutput().getKinesisDataStream().getArn());
        System.out.println("RoleArn - " + describeStreamProcessorResult.getRoleArn());
        System.out.println(
                "CollectionId - " +
describeStreamProcessorResult.getSettings().getFaceSearch().getCollectionId());
```

```
        System.out.println("Status - " + describeStreamProcessorResult.getStatus());
        System.out.println("Status message - " +
 describeStreamProcessorResult.getStatusMessage());
        System.out.println("Creation timestamp - " +
 describeStreamProcessorResult.getCreationTimestamp());
        System.out.println("Last update timestamp - " +
 describeStreamProcessorResult.getLastUpdateTimestamp());
    }

    public void listStreamProcessors() {
        ListStreamProcessorsResult listStreamProcessorsResult =
                rekognitionClient.listStreamProcessors(new
 ListStreamProcessorsRequest().withMaxResults(100));

        //List all stream processors (and state) returned from Rekognition
        for (StreamProcessor streamProcessor :
 listStreamProcessorsResult.getStreamProcessors()) {
            System.out.println("StreamProcessor name - " + streamProcessor.getName());
            System.out.println("Status - " + streamProcessor.getStatus());
        }
    }
}
```

Lectura de los resultados del análisis de vídeo en streaming

Puede utilizar la biblioteca de clientes de Amazon Kinesis Data Streams para consumir los resultados de análisis que se envían a la transmisión de salida de Amazon Kinesis Data Streams. Para obtener más información, consulte Reading Data from a Kinesis Data Stream. Amazon Rekognition Video coloca un registro de fotograma de JSON para cada fotograma analizado en la transmisión de salida de Kinesis. Amazon Rekognition Video no analiza todos los fotogramas que recibe a través de la transmisión de vídeo de Kinesis.

Un registro de fotograma que se envía a un flujo de datos de Kinesis contiene información acerca del fragmento de flujo de datos de Kinesis en el que está el fotograma, donde está el fotograma en el fragmento y los rostros reconocidos en el fotograma. También incluye información de estado para el procesador de streaming. Para obtener más información, consulte Descripción del registro de fotogramas de JSON para reconocimiento facial de Kinesis.

Amazon Kinesis Video Streams Parser Library contiene ejemplos de pruebas que consumen los resultados de Amazon Rekognition Video y los integra con la transmisión de vídeo original de

Kinesis. Para obtener más información, consulte Visualización local de los resultados de Rekognition con Kinesis Video Streams.

Amazon Rekognition Video transmite la información de análisis de Amazon Rekognition Video a los flujos de datos de Kinesis. A continuación, se muestra un ejemplo de JSON para un registro único.


```
{
  "InputInformation": {
    "KinesisVideo": {
      "StreamArn": "arn:aws:kinesisvideo:us-west-2:nnnnnnnnnnnn:stream/stream-name",
      "FragmentNumber": "91343852333289682796718532614445757584843717598",
      "ServerTimestamp": 1510552593.455,
      "ProducerTimestamp": 1510552593.193,
      "FrameOffsetInSeconds": 2
    }
  },
  "StreamProcessorInformation": {
    "Status": "RUNNING"
  },
  "FaceSearchResponse": [
    {
      "DetectedFace": {
        "BoundingBox": {
          "Height": 0.075,
          "Width": 0.05625,
          "Left": 0.428125,
          "Top": 0.40833333
        },
        "Confidence": 99.975174,
        "Landmarks": [
          {
            "X": 0.4452057,
            "Y": 0.4395594,
            "Type": "eyeLeft"
          },
          {
            "X": 0.46340984,
            "Y": 0.43744427,
            "Type": "eyeRight"
          },
          {
            "X": 0.45960626,
            "Y": 0.4526856,
```

```
            "Type": "nose"
          },
          {
            "X": 0.44958648,
            "Y": 0.4696949,
            "Type": "mouthLeft"
          },
          {
            "X": 0.46409217,
            "Y": 0.46704912,
            "Type": "mouthRight"
          }
        ],
        "Pose": {
          "Pitch": 2.9691637,
          "Roll": -6.8904796,
          "Yaw": 23.84388
        },
        "Quality": {
          "Brightness": 40.592964,
          "Sharpness": 96.09616
        }
      },
      "MatchedFaces": [
        {
          "Similarity": 88.863960,
          "Face": {
            "BoundingBox": {
              "Height": 0.557692,
              "Width": 0.749838,
              "Left": 0.103426,
              "Top": 0.206731
            },
            "FaceId": "ed1b560f-d6af-5158-989a-ff586c931545",
            "Confidence": 99.999201,
            "ImageId": "70e09693-2114-57e1-807c-50b6d61fa4dc",
            "ExternalImageId": "matchedImage.jpeg"
          }
        }
      ]
    }
  ]
}
```

En el ejemplo de JSON, observe lo siguiente:

- InputInformation— Información sobre la transmisión de vídeo de Kinesis que se utiliza para transmitir vídeo a Amazon Rekognition Video. Para obtener más información, consulte InputInformation.
- StreamProcessorInformation— Información de estado del procesador de transmisión de vídeo Amazon Rekognition Video. El único valor posible para el campo `Status` es RUNNING. Para obtener más información, consulte StreamProcessorInformation.
- FaceSearchResponse— Contiene información sobre los rostros del vídeo en streaming que coinciden con los rostros de la colección de entrada. FaceSearchResponsecontiene un DetectedFace objeto, que es un rostro que se detectó en el fotograma de vídeo analizado. Para cada rostro detectado, la matriz `MatchedFaces` contiene un matriz de objetos de rostro coincidentes (MatchedFace) encontrados en la colección de entrada, junto con una puntuación de similitud.

Asignación de la secuencia de vídeo de Kinesis al flujo de datos de Kinesis

Es posible que desee asignar los fotogramas de transmisión de vídeo de Kinesis a los fotogramas analizados que se envían a la flujo de datos de Kinesis. Por ejemplo, durante la visualización de un vídeo de streaming, es posible que desee mostrar cuadros alrededor de los rostros de las personas reconocidas. Las coordenadas del cuadro delimitador se envían como parte del registro de reconocimiento facial de Kinesis al flujo de datos de Kinesis. Para mostrar el cuadro delimitador correctamente, debe asignar la información temporal que se envía con el registro de reconocimiento facial de Kinesis con los fotogramas correspondientes a la transmisión de vídeo de Kinesis de origen.

La técnica que utiliza para asignar la transmisión de vídeo de Kinesis a la flujo de datos de Kinesis depende de si va a transmitir medios en directo (como un vídeo de streaming en directo) o si va a transmitir medios archivados (como un vídeo almacenado).

Asignación cuando se transmiten medios en directo

Para asignar un fotograma de transmisión de vídeo de Kinesis a un fotograma de flujo de datos de Kinesis

1. Defina el parámetro `FragmentTimeCodeType` de entrada de la PutMediaoperación en`RELATIVE`.
2. Llame a `PutMedia` para enviar contenido en vivo a la transmisión de vídeo de Kinesis.

3. Cuando reciba un registro del reconocimiento facial de Kinesis de los flujos de datos de Kinesis, almacene los valores de `ProducerTimestamp` y `FrameOffsetInSeconds` del campo KinesisVideo.
4. Calcule la marca temporal correspondiente al fotograma de transmisión de vídeo de Kinesis sumando los valores del campo `ProducerTimestamp` y `FrameOffsetInSeconds`.

Asignación cuando se transmiten medios archivados

Para asignar un fotograma de transmisión de vídeo de Kinesis a un fotograma de flujo de datos de Kinesis

1. Llame PutMediapara enviar contenido multimedia archivado a la transmisión de vídeo de Kinesis.
2. Cuando reciba un objeto `Acknowledgement` de la respuesta de la operación `PutMedia`, almacene el valor del campo `FragmentNumber` del campo Payload. `FragmentNumber` es el número de fragmento del clúster de MKV.
3. Cuando reciba un registro del reconocimiento facial de Kinesis de los flujos de datos de Kinesis, almacene el valor del campo `FrameOffsetInSeconds` del campo KinesisVideo.
4. Calcule el mapeo utilizando los valores de `FrameOffsetInSeconds` y `FragmentNumber` almacenados en los pasos 2 y 3. `FrameOffsetInSeconds` es la diferencia del fragmento con el `FragmentNumber` enviado al flujo de datos de Amazon Kinesis. Para obtener más información sobre cómo obtener los fotogramas de vídeo para un número de fragmento determinado, consulte Medios archivados de Amazon Kinesis Video Streams.

Visualización local de los resultados de Rekognition con Kinesis Video Streams

Puede ver los resultados de Amazon Rekognition Video que se muestran en su feed de Amazon Kinesis Video Streams mediante las pruebas de ejemplo de la biblioteca de analizadores de Amazon Kinesis Video Streams que se proporcionan en Rekognition Examples. KinesisVideo `KinesisVideoRekognitionIntegrationExample`Muestra cuadros delimitadores sobre los rostros detectados y renderiza el vídeo de forma local. JFrame Este proceso supone que ha conectado correctamente una entrada multimedia de la cámara de un dispositivo a una transmisión de vídeo de Kinesis y ha iniciado un procesador de transmisión de Amazon Rekognition. Para obtener más información, consulte Transmisión mediante un GStreamer complemento.

Paso 1: Instalación de Kinesis Video Streams Parser Library

Para crear un directorio y descargar el repositorio de GitHub, ejecute el siguiente comando:

```
$ git clone https://github.com/aws/amazon-kinesis-video-streams-parser-library.git
```

Navegue hasta el directorio de la biblioteca y ejecute el siguiente comando de Maven para realizar una instalación limpia:

```
$ mvn clean install
```

Paso 2: Ejemplo de configuración de la prueba de integración de Kinesis Video Streams y Rekognition

Abra el archivo `KinesisVideoRekognitionIntegrationExampleTest.java`. Elimine `@Ignore` justo después del encabezado de la clase. Rellene los campos de datos con la información de sus recursos de Amazon Kinesis y Amazon Rekognition. Para obtener más información, consulte Configuración de los recursos de Amazon Rekognition Video y Amazon Kinesis. Si está transmitiendo vídeo a su transmisión de vídeo de Kinesis, elimine el parámetro `inputStream`.

Consulte el siguiente ejemplo de código:

```
RekognitionInput rekognitionInput = RekognitionInput.builder()
  .kinesisVideoStreamArn("arn:aws:kinesisvideo:us-east-1:123456789012:stream/
rekognition-test-video-stream")
  .kinesisDataStreamArn("arn:aws:kinesis:us-east-1:123456789012:stream/
AmazonRekognition-rekognition-test-data-stream")
  .streamingProcessorName("rekognition-test-stream-processor")
  // Refer how to add face collection :
  // https://docs.aws.amazon.com/rekognition/latest/dg/add-faces-to-collection-
procedure.html
  .faceCollectionId("rekognition-test-face-collection")
  .iamRoleArn("rekognition-test-IAM-role")
  .matchThreshold(0.95f)
  .build();

KinesisVideoRekognitionIntegrationExample example =
 KinesisVideoRekognitionIntegrationExample.builder()
  .region(Regions.US_EAST_1)
  .kvsStreamName("rekognition-test-video-stream")
```

```
  .kdsStreamName("AmazonRekognition-rekognition-test-data-stream")
  .rekognitionInput(rekognitionInput)
  .credentialsProvider(new ProfileCredentialsProvider())
  // NOTE: Comment out or delete the inputStream parameter if you are streaming video,
otherwise
  // the test will use a sample video.
  //.inputStream(TestResourceUtil.getTestInputStream("bezos_vogels.mkv"))
  .build();
```

Paso 3: Ejemplo de ejecución de la prueba de integración de Kinesis Video Streams y Rekognition

Asegúrese de que su transmisión de vídeo de Kinesis reciba entradas multimedia si está transmitiendo a ella y comience a analizar la transmisión con Amazon Rekognition Video Stream Processor en ejecución. Para obtener más información, consulte Descripción general de las operaciones del procesador de transmisión de Amazon Rekognition Video. Organiza la `KinesisVideoRekognitionIntegrationExampleTest` clase como una JUnit prueba. Tras un breve retraso, se abre una nueva ventana con una transmisión de vídeo de la transmisión de vídeo de Kinesis con cuadros delimitadores dibujados sobre las caras detectadas.

> **Note**
>
> Las caras de la colección utilizada en este ejemplo deben tener un identificador de imagen externo (el nombre del archivo) especificado en este formato para que las etiquetas de los cuadros delimitadores muestren texto significativo: PersonName 1 de confianza, PersonName 2 intrusos, 3 neutrales, etc. PersonName Las etiquetas también se pueden codificar por colores y se pueden personalizar en el archivo.java. FaceType

Descripción del registro de fotogramas de JSON para reconocimiento facial de Kinesis

Amazon Rekognition Video puede reconocer rostros en una transmisión de vídeo. Amazon Rekognition Video coloca un registro de fotograma de JSON para cada fotograma analizado en el flujo de datos de Kinesis. Amazon Rekognition Video no analiza todos los fotogramas que recibe a través de la transmisión de vídeo de Kinesis.

El registro de fotograma de JSON contiene información acerca del streaming de entrada y de salida, el estado del procesador de streaming e información acerca de rostros que se han reconocido en el fotograma analizado. Esta sección contiene información de referencia para el registro de fotogramas de JSON.

La siguiente es la sintaxis de JSON para un registro de flujo de datos de Kinesis. Para obtener más información, consulte Trabajar con eventos de vídeo en streaming.

> **Note**
>
> La API de Amazon Rekognition Video funciona comparando los rostros de la secuencia de entrada con una colección de caras y devolviendo las coincidencias más próximas que se encuentren con una puntuación de similitud.

```
{
    "InputInformation": {
        "KinesisVideo": {
            "StreamArn": "string",
            "FragmentNumber": "string",
            "ProducerTimestamp": number,
            "ServerTimestamp": number,
            "FrameOffsetInSeconds": number
        }
    },
    "StreamProcessorInformation": {
        "Status": "RUNNING"
    },
    "FaceSearchResponse": [
        {
            "DetectedFace": {
                "BoundingBox": {
                    "Width": number,
                    "Top": number,
                    "Height": number,
                    "Left": number
                },
                "Confidence": number,
                "Landmarks": [
                    {
                        "Type": "string",
                        "X": number,
                        "Y": number
                    }
                ],
                "Pose": {
                    "Pitch": number,
```

```
                    "Roll": number,
                    "Yaw": number
                },
                "Quality": {
                    "Brightness": number,
                    "Sharpness": number
                }
            },
            "MatchedFaces": [
                {
                    "Similarity": number,
                    "Face": {
                        "BoundingBox": {
                            "Width": number,
                            "Top": number,
                            "Height": number,
                            "Left": number
                        },
                        "Confidence": number,
                        "ExternalImageId": "string",
                        "FaceId": "string",
                        "ImageId": "string"
                    }
                }
            ]
        }
    ]
}
```

Registro de JSON

El registro de JSON incluye información sobre un fotograma que ha procesado Amazon Rekognition Video. El registro incluye información acerca del vídeo de streaming, el estado del fotograma analizado e información acerca de rostros que se han reconocido en el fotograma.

InputInformation

Información sobre la transmisión de vídeo de Kinesis que se utiliza para transmitir vídeo en Amazon Rekognition Video.

Tipo: objeto InputInformation

StreamProcessorInformation

Información acerca del procesador de streaming de Amazon Rekognition Video. Esto incluye información de estado para el estado actual del procesador de streaming.

Tipo: objeto StreamProcessorInformation

FaceSearchResponse

Información acerca de los rostros detectados en un fotograma de vídeo en streaming y los rostros coincidentes encontrados en la colección de entrada.

Tipo: matriz de objetos FaceSearchResponse

InputInformation

Información acerca de una transmisión de vídeo de origen que utiliza Amazon Rekognition Video. Para obtener más información, consulte Trabajar con eventos de vídeo en streaming.

KinesisVideo

Tipo: objeto KinesisVideo

KinesisVideo

Información sobre la transmisión de vídeo de Kinesis que transmite el vídeo de origen a Amazon Rekognition Video. Para obtener más información, consulte Trabajar con eventos de vídeo en streaming.

StreamArn

El nombre de recurso de Amazon (ARN) de la transmisión de vídeo de Kinesis.

Tipo: cadena

FragmentNumber

El fragmento del vídeo en streaming que contiene el fotograma que representa este registro.

Tipo: cadena

ProducerTimestamp

La marca temporal Unix del lado del productor del fragmento. Para obtener más información, consulte PutMedia.

Tipo: Number

ServerTimestamp

La marca temporal Unix del lado del servidor del fragmento. Para obtener más información, consulte PutMedia.

Tipo: Number

FrameOffsetInSeconds

El desfase del fotograma (en segundos) dentro del fragmento.

Tipo: Number

StreamProcessorInformation

Información de estado acerca del procesador de streaming.

Estado

El estado actual del procesador de streaming. El único valor posible es RUNNING.

Tipo: cadena

FaceSearchResponse

Información acerca de un rostro detectado en un fotograma de vídeo en streaming y los rostros de una colección que coinciden con el rostro detectado. Especifica la colección en una llamada a CreateStreamProcessor. Para obtener más información, consulte Trabajar con eventos de vídeo en streaming.

DetectedFace

Detalles de un rostro detectado en un fotograma de vídeo analizado.

Tipo: objeto DetectedFace

MatchedFaces

Una matriz de detalles de rostros en una colección que coincide con el rostro detectado en `DetectedFace`.

Tipo: matriz de objetos MatchedFace

DetectedFace

Información sobre un rostro que se detectó en un fotograma de vídeo en streaming. Los rostros coincidentes en la colección de entrada están disponibles en campo de objeto MatchedFace.

BoundingBox

Las coordenadas del cuadro delimitador de un rostro que se ha detectado dentro de un fotograma de vídeo analizado. El BoundingBox objeto tiene las mismas propiedades que el BoundingBox objeto que se utiliza para el análisis de imágenes.

Tipo: objeto BoundingBox

Confianza

El nivel de confianza (de 1 a 100) que tiene Amazon Rekognition Video de que el rostro detectado es en realidad un rostro. 1 es la confianza más baja y 100 es la más alta.

Tipo: Number

Referencias

Una matriz de referencias faciales.

Tipo: matriz de objetos Landmark

Postura

Indica la postura del rostro tal como determina su cabeceo, balanceo y desviación.

Tipo: Objeto de Pose

Calidad

Identifica el brillo y la nitidez de la imagen del rostro.

Tipo: objeto ImageQuality

MatchedFace

Información sobre un rostro que coincide con un rostro detectado en un fotograma de vídeo analizado.

Rostro

Información de coincidencia de rostro para un rostro en la colección de entrada que coincide con el rostro en el objeto DetectedFace.

Tipo: objeto Face

Similitud

El nivel de confianza (de 1 a 100) con el que coinciden las caras. 1 es la confianza más baja y 100 es la más alta.

Tipo: Number

Transmisión mediante un GStreamer complemento

Amazon Rekognition Video puede analizar una transmisión de vídeo en directo desde la cámara de un dispositivo. Para acceder a la entrada multimedia desde la fuente de un dispositivo, es necesario instalarlo GStreamer. GStreamer es un software de marco multimedia de terceros que conecta las fuentes multimedia y las herramientas de procesamiento en los flujos de trabajo. También debe instalar el complemento Amazon Kinesis Video Streams Producer para Gstreamer. En este proceso se presupone que ha configurado correctamente los recursos de Amazon Rekognition Video y Amazon Kinesis. Para obtener más información, consulte Configuración de los recursos de Amazon Rekognition Video y Amazon Kinesis.

Paso 1: Instale Gstreamer

Descargue e instale Gstreamer, un software de plataforma multimedia de terceros. Puede usar un software de administración de paquetes como Homebrew (Gstreamer en Homebrew) u obtenerlo directamente desde el sitio web de Freedesktop.

Compruebe que la instalación de Gstreamer se ha realizado correctamente iniciando una transmisión de vídeo con una fuente de prueba desde su terminal de línea de comandos.

```
$ gst-launch-1.0 videotestsrc ! autovideosink
```

Paso 2: Instale el complemento Kinesis Video Streams Producer

En esta sección, descargará la biblioteca de Amazon Kinesis Video Streams Producer e instalará el complemento Kinesis Video Streams Gstreamer.

Cree un directorio y clone el código fuente del repositorio GitHub. Asegúrese de incluir el parámetro `--recursive`.

```
$ git clone --recursive https://github.com/awslabs/amazon-kinesis-video-streams-
producer-sdk-cpp.git
```

Siga las instrucciones de la biblioteca para configurar y compilar el proyecto. Asegúrese de utilizar los comandos específicos de la plataforma para su sistema operativo. Utilice el parámetro `-DBUILD_GSTREAMER_PLUGIN=ON` cuando ejecute `cmake` para instalar el complemento Kinesis Video Streams Gstreamer. Este proyecto requiere los siguientes paquetes adicionales que se incluyen en la instalación: GCC o Clang, Curl, Openssl y Log4cplus. Si la compilación falla porque falta un paquete, compruebe que el paquete esté instalado y en su ruta. Si se produce el error «No se puede ejecutar el programa compilado en C» durante la compilación, vuelva a ejecutar el comando build. A veces, no se encuentra el compilador de C correcto.

Verifique la instalación del complemento Kinesis Video Streams mediante la ejecución del siguiente comando.

```
$ gst-inspect-1.0 kvssink
```

Debería aparecer la siguiente información, como los detalles de fábrica y del plugin:

```


Factory Details:
  Rank                     primary + 10 (266)
  Long-name                KVS Sink
  Klass                    Sink/Video/Network
  Description              GStreamer AWS KVS plugin
  Author                   AWS KVS <kinesis-video-support@amazon.com>

Plugin Details:
  Name                     kvssink
  Description              GStreamer AWS KVS plugin
  Filename                 /Users/YOUR_USER/amazon-kinesis-video-streams-producer-sdk-
cpp/build/libgstkvssink.so
  Version                  1.0
  License                  Proprietary
  Source module            kvssinkpackage
  Binary package           GStreamer
  Origin URL               http://gstreamer.net/

  ...
```

Paso 3: Ejecute Gstreamer con el complemento Kinesis Video Streams

Antes de empezar a transmitir desde la cámara de un dispositivo a Kinesis Video Streams, puede que necesite convertir la fuente multimedia en un códec aceptable para Kinesis Video Streams. Para determinar las especificaciones y las capacidades de formato de los dispositivos actualmente conectados a su máquina, ejecute el siguiente comando.

```
$ gst-device-monitor-1.0
```

Para iniciar la transmisión, inicie Gstreamer con el siguiente comando de ejemplo y añada sus credenciales y la información de Amazon Kinesis Video Streams. Debe usar las claves de acceso y la región del rol de servicio de IAM que creó al conceder a Amazon Rekognition acceso a sus transmisiones de Kinesis. Para obtener más información acerca de las claves de acceso, consulte Administración de las claves de acceso de los usuarios de IAM en la Guía del usuario de IAM. Además, puede ajustar los parámetros de los argumentos del formato de vídeo según lo requiera su uso y estén disponibles en su dispositivo.

```
$ gst-launch-1.0 autovideosrc device=/dev/video0 ! videoconvert ! video/x-
raw,format=I420,width=640,height=480,framerate=30/1 !
                x264enc bframes=0 key-int-max=45 bitrate=500 ! video/x-h264,stream-
format=avc,alignment=au,profile=baseline !
                kvssink stream-name="YOUR_STREAM_NAME" storage-size=512 access-
key="YOUR_ACCESS_KEY" secret-key="YOUR_SECRET_ACCESS_KEY" aws-region="YOUR_AWS_REGION"
```

Para obtener más comandos de inicio, consulte Ejemplos de comandos de GStreamer lanzamiento.

> **Note**
>
> Si el comando de lanzamiento termina con un error no relacionado con la negociación, compruebe el resultado del monitor de dispositivos y asegúrese de que los valores del parámetro `videoconvert` corresponden a las capacidades válidas de su dispositivo.

Verá una transmisión de vídeo de la cámara de su dispositivo en la transmisión de vídeo de Kinesis después de unos segundos. Para empezar a detectar y comparar rostros con Amazon Rekognition, inicie el procesador de transmisión Amazon Rekognition Video. Para obtener más información,

consulte Descripción general de las operaciones del procesador de transmisión de Amazon Rekognition Video.

Solución de problemas de vídeo en streaming

Este tema ofrece información sobre cómo solucionar problemas en el uso de Amazon Rekognition Video con vídeos en streaming.

Temas

- No sé si mi procesador de streaming se ha creado correctamente
- No sé si he configurado correctamente mi procesador de streaming
- Mi procesador de streaming no está devolviendo resultados
- El estado de mi procesador de streaming es FAILED
- Mi procesador de streaming no está devolviendo los resultados esperados

No sé si mi procesador de streaming se ha creado correctamente

Utilice el siguiente AWS CLI comando para obtener una lista de los procesadores de flujo y su estado actual.

```
aws rekognition list-stream-processors
```

Puede obtener detalles adicionales mediante el siguiente AWS CLI comando. Reemplace `stream-processor-name` por el nombre del procesador de streaming necesario.

```
aws rekognition describe-stream-processor --name stream-processor-name
```

No sé si he configurado correctamente mi procesador de streaming

Si el código no está devolviendo los resultados de análisis de Amazon Rekognition Video, es posible que su procesador de streaming no esté configurado correctamente. Realice lo siguiente para confirmar que su procesador de streaming se ha configurado correctamente y que puede producir resultados.

Para determinar si su solución está configurada correctamente

1. Ejecute el siguiente comando para confirmar que el procesador de streaming se encuentra en estado de ejecución. Cambie `stream-processor-name` por el nombre de su procesador de

streaming. El procesador de streaming está en ejecución si el valor de `Status` es `RUNNING`. Si el estado es `RUNNING` y no está obteniendo resultados, consulte Mi procesador de streaming no está devolviendo resultados. Si el estado es `FAILED`, consulte El estado de mi procesador de streaming es FAILED.

```
aws rekognition describe-stream-processor --name stream-processor-name
```

2. Si su procesador de transmisiones está en ejecución, ejecute el siguiente PowerShell comando o Bash para leer los datos de la transmisión de datos de Kinesis de salida.

   Bash

   ```
   SHARD_ITERATOR=$(aws kinesis get-shard-iterator --shard-id shardId-000000000000
    --shard-iterator-type TRIM_HORIZON --stream-name kinesis-data-stream-name --query
    'ShardIterator')
                           aws kinesis get-records --shard-iterator $SHARD_ITERATOR
   ```

   PowerShell

   ```
   aws kinesis get-records --shard-iterator ((aws kinesis get-shard-iterator --shard-
   id shardId-000000000000 --shard-iterator-type TRIM_HORIZON --stream-name kinesis-
   data-stream-name).split('"')[4])
   ```

3. Utilice la herramienta Decode en el sitio web de Base64 Decode para decodificar el resultado en una cadena legible para las personas. Para obtener más información, consulte el Paso 3: Obtener el registro.
4. Si los comandos funcionan y ve resultados de detección de rostros en la secuencia de datos de Kinesis, la solución está configurada correctamente. Si el comando da error, compruebe las otras sugerencias de solución de problemas y consulte Otorgar a Amazon Rekognition Video acceso a sus recursos.

Como alternativa, puede usar el AWS Lambda plano «kinesis-process-record» para registrar los mensajes desde la transmisión de datos de Kinesis CloudWatch para una visualización continua. Esto implica costos adicionales de y. AWS Lambda CloudWatch

Mi procesador de streaming no está devolviendo resultados

Su procesador de streaming podría no devolver resultados por varios motivos.

Motivo 1: su procesador de streaming no está configurado correctamente

Su procesador de streaming podría no estar configurado correctamente. Para obtener más información, consulte No sé si he configurado correctamente mi procesador de streaming.

Motivo 2: Su procesador de streaming no está en el estado RUNNING

Para solucionar el estado de un procesador de streaming

1. Compruebe el estado del procesador de flujo con el siguiente AWS CLI comando.

```
aws rekognition describe-stream-processor --name stream-processor-name
```

2. Si el valor de `Status` es `STOPPED`, inicie el procesador de streaming con el siguiente comando:

```
aws rekognition start-stream-processor --name stream-processor-name
```

3. Si el valor de `Status` es `FAILED`, consulte El estado de mi procesador de streaming es FAILED.
4. Si el valor de `Status` es `STARTING`, espere 2 minutos y compruebe el estado repitiendo el paso 1. Si el valor de Status sigue siendo `STARTING`, realice lo siguiente:
   a. Elimine el procesador de streaming con el siguiente comando.

   ```
   aws rekognition delete-stream-processor --name stream-processor-name
   ```

   b. Cree un nuevo procesador de streaming con la misma configuración. Para obtener más información, consulte Trabajar con eventos de vídeo en streaming.
   c. Si sigues teniendo problemas, ponte en contacto con AWS Support.
5. Si el valor de `Status` es `RUNNING`, consulte Motivo 3: no hay datos activos en la transmisión de vídeo de Kinesis.

Motivo 3: no hay datos activos en la transmisión de vídeo de Kinesis

Para comprobar si hay datos activos en la transmisión de vídeo de Kinesis

1. Inicie sesión en la Consola de administración de AWS consola https://console.aws.amazon.com/kinesisvideo/de Amazon Kinesis Video Streams y ábrala en.

2. Seleccione la transmisión de vídeo de Kinesis que es la entrada para el procesador de transmisión de Amazon Rekognition.
3. Si la vista previa indica Ningún dato en la transmisión, significa que no hay ningún dato en la transmisión de entrada para que lo procese Amazon Rekognition Video.

Para obtener información sobre la producción de vídeo con Kinesis Video Streams, consulte Kinesis Video Streams Producer Libraries.

El estado de mi procesador de streaming es FAILED

Puede comprobar el estado de un procesador de streaming mediante el siguiente AWS CLI comando.

```
aws rekognition describe-stream-processor --name stream-processor-name
```

Si el valor de Status es FAILED, compruebe la información de solución de problemas de los siguientes mensajes de error.

Error: «Acceso denegado al rol»

El rol de IAM que utiliza el procesador de streaming no existe o Amazon Rekognition Video no tiene permiso para asumir el rol.

Para solucionar problemas de acceso con el rol de IAM

1. Inicie sesión en la consola de IAM Consola de administración de AWS y ábrala en https://console.aws.amazon.com/iam/.
2. En el panel de navegación izquierdo, elija Roles y confirme que el rol existe.
3. Si el rol existe, compruebe que el rol tenga la política de AmazonRekognitionServiceRolepermisos.
4. Si el rol no existe o no tiene los permiso adecuados, consulte Otorgar a Amazon Rekognition Video acceso a sus recursos.
5. Inicie el procesador de transmisión con el siguiente AWS CLI comando.

   ```
   aws rekognition start-stream-processor --name stream-processor-name
   ```

Error: "Access denied to Kinesis Video (Acceso denegado a vídeo de Kinesis) o Access denied to Kinesis Data (Acceso denegado a datos de Kinesis)"

El rol no tiene acceso a las operaciones de API de Kinesis Video Streams `GetMedia` y `GetDataEndpoint`. Es posible que tampoco tenga acceso a las operaciones de API de Kinesis Data Streams `PutRecord` y `PutRecords`.

Solución de problemas de permisos de API

1. Inicie sesión en la consola de IAM Consola de administración de AWS y ábrala en https://console.aws.amazon.com/iam/.
2. Abra el rol y asegúrese de que tiene la siguiente política de permisos asociada.
3. Si falta alguno de los permisos, actualice la política. Para obtener más información, consulte Otorgar a Amazon Rekognition Video acceso a sus recursos.

Error: «La transmisión ***input-video-stream-name*** no existe»

La entrada de transmisión de vídeo de Kinesis para el procesador de streaming no existe o no está configurada correctamente.

Para solucionar problemas de la secuencia de vídeo de Kinesis

1. Utilice el siguiente comando para confirmar que la secuencia existe.

   ```
   aws kinesisvideo list-streams
   ```

2. Si la secuencia existe, compruebe lo siguiente.

   - El nombre de recurso de Amazon (ARN) es el mismo que el ARN de la secuencia de entrada del procesador de streaming.
   - La secuencia de vídeo de Kinesis debe encontrarse en la misma Región que el procesador de streaming.

   Si el procesador de transmisión no está configurado correctamente, elimínelo con el siguiente AWS CLI comando.

   ```
   aws rekognition delete-stream-processor --name stream-processor-name
   ```

3. Cree un nuevo procesador de streaming con la secuencia de vídeo de Kinesis prevista. Para obtener más información, consulte Creación del procesador de flujo de búsqueda de rostros de Amazon Rekognition Video.

Error: «Colección no encontrada»

La colección de Amazon Rekognition que utiliza el procesador de streaming para comparar rostros no existe, o se está utilizando la colección equivocada.

Para confirmar la colección

1. Use el siguiente AWS CLI comando para determinar si existe la colección requerida. Cambie `region` a la AWS región en la que está ejecutando el procesador de streaming.

   ```
   aws rekognition list-collections --region region
   ```

   Si la colección necesaria no existe, cree una nueva colección y añada información de rostros. Para obtener más información, consulte Búsqueda de rostros en una colección.
2. En la llamada a CreateStreamProcessor, compruebe que el valor del parámetro de entrada de `CollectionId` es correcto.
3. Inicie el procesador de transmisión con el siguiente AWS CLI comando.

   ```
   aws rekognition start-stream-processor --name stream-processor-name
   ```

Error: «***account-id***No se encontró ***output-kinesis-data-stream-name*** la transmisión en la cuenta»

La transmisión de datos de Kinesis de salida que utiliza el procesador de transmisiones no existe Cuenta de AWS o no se encuentra en la misma AWS región que su procesador de transmisiones.

Para solucionar problemas de la secuencia de datos de Kinesis

1. Utilice el siguiente AWS CLI comando para determinar si existe la transmisión de datos de Kinesis. Cambie `region` a la AWS región en la que utiliza el procesador de transmisiones.

   ```
   aws kinesis list-streams --region region
   ```

2. Si la secuencia de datos de Kinesis existe, compruebe que el nombre de la secuencia de datos de Kinesis es el mismo que el nombre de la secuencia de salida que utiliza el procesador de streaming.
3. Si la transmisión de datos de Kinesis no existe, puede que esté en otra AWS región. La secuencia de datos de Kinesis debe encontrarse en la misma Región que el procesador de streaming.
4. Si es necesario, cree una nueva secuencia de datos de Kinesis.

    a. Cree una secuencia de datos de Kinesis con el mismo nombre que el usado por el procesador de streaming. Para obtener más información, consulte Paso 1: Crear una secuencia de datos.

    b. Inicie el procesador de transmisiones con el siguiente AWS CLI comando.

```
aws rekognition start-stream-processor --name stream-processor-name
```

Mi procesador de streaming no está devolviendo los resultados esperados

Si el procesador de streaming no está devolviendo los rostros coincidentes esperados, utilice la siguiente información.

- Búsqueda de rostros en una colección
- Recomendaciones de configuración de la cámara (vídeo en streaming)

Recorridos de las personas

> **Note**
>
> Aviso de fin de soporte: el 31 de octubre de 2025, AWS dejará de ofrecer soporte para la característica Recorridos de las personas de Amazon Rekognition. Después del 31 de octubre de 2025, ya no podrá usar la característica Recorridos de las personas de Rekognition. Para obtener más información, visite esta publicación del blog.

Amazon Rekognition Video puede hacer un seguimiento del recorrido que siguen las personas en los vídeos y proporcionar información como:

- La ubicación de las personas en un fotograma de vídeo en el momento en que se realiza el seguimiento de su recorrido.
- Referencias faciales como, por ejemplo, la posición del ojo izquierdo, cuando se detectaron.

El recorrido de las personas de Amazon Rekognition Video en vídeos almacenados es una operación asíncrona. Para iniciar la búsqueda de rutas de las personas en los vídeos, llame a StartPersonTracking. Amazon Rekognition Video publica el estado de finalización de una operación de análisis de vídeo en un tema de Amazon Simple Notification Service. Si el análisis de vídeo es correcto, puede llamar a GetPersonTracking para obtener los resultados. Para obtener más información sobre cómo llamar a las operaciones de la API de Amazon Rekognition Video consulte Cómo llamar a las operaciones de Amazon Rekognition Video.

El siguiente procedimiento muestra cómo realizar un seguimiento del recorrido de las personas en un vídeo almacenado en un bucket de Amazon S3. El ejemplo amplía el código en Análisis de un vídeo almacenado en un bucket de Amazon S3 con Java o Python (SDK) que utiliza una cola de Amazon Simple Queue Service para obtener el estado de realización de una solicitud de análisis de vídeo.

Para detectar personas en un vídeo almacenado en un bucket de Amazon S3 (SDK)

1. Realice Análisis de un vídeo almacenado en un bucket de Amazon S3 con Java o Python (SDK).
2. Añada el código siguiente a la clase `VideoDetect` que ha creado en el paso 1.

Java

```
      //Copyright 2018 Amazon.com, Inc. or its affiliates. All Rights
 Reserved.
      //PDX-License-Identifier: MIT-0 (For details, see https://github.com/
awsdocs/amazon-rekognition-developer-guide/blob/master/LICENSE-SAMPLECODE.)

       //
Persons~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
       private static void StartPersonDetection(String bucket, String video)
 throws Exception{


           NotificationChannel channel= new NotificationChannel()
                   .withSNSTopicArn(snsTopicArn)
                   .withRoleArn(roleArn);

        StartPersonTrackingRequest req = new StartPersonTrackingRequest()
                .withVideo(new Video()
                        .withS3Object(new S3Object()
                            .withBucket(bucket)
                            .withName(video)))
                .withNotificationChannel(channel);



        StartPersonTrackingResult startPersonDetectionResult =
 rek.startPersonTracking(req);
        startJobId=startPersonDetectionResult.getJobId();

       }

       private static void GetPersonDetectionResults() throws Exception{
           int maxResults=10;
           String paginationToken=null;
           GetPersonTrackingResult personTrackingResult=null;

           do{
               if (personTrackingResult !=null){
                   paginationToken = personTrackingResult.getNextToken();
               }
```

```
                personTrackingResult = rek.getPersonTracking(new
GetPersonTrackingRequest()
                     .withJobId(startJobId)
                     .withNextToken(paginationToken)
                     .withSortBy(PersonTrackingSortBy.TIMESTAMP)
                     .withMaxResults(maxResults));

                VideoMetadata
videoMetaData=personTrackingResult.getVideoMetadata();

                System.out.println("Format: " + videoMetaData.getFormat());
                System.out.println("Codec: " + videoMetaData.getCodec());
                System.out.println("Duration: " +
videoMetaData.getDurationMillis());
                System.out.println("FrameRate: " +
videoMetaData.getFrameRate());


                //Show persons, confidence and detection times
                List<PersonDetection> detectedPersons=
personTrackingResult.getPersons();

                for (PersonDetection detectedPerson: detectedPersons) {

                   long seconds=detectedPerson.getTimestamp()/1000;
                   System.out.print("Sec: " + Long.toString(seconds) + " ");
                   System.out.println("Person Identifier: "  +
detectedPerson.getPerson().getIndex());
                        System.out.println();
                }
           }  while (personTrackingResult !=null &&
personTrackingResult.getNextToken() != null);

       }
```

En la función `main`, reemplace las líneas:

```
       StartLabelDetection(amzn-s3-demo-bucket, video);

       if (GetSQSMessageSuccess()==true)
        GetLabelDetectionResults();
```

por:

```
        StartPersonDetection(amzn-s3-demo-bucket, video);

        if (GetSQSMessageSuccess()==true)
         GetPersonDetectionResults();
```

Java V2

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.S3Object;
import software.amazon.awssdk.services.rekognition.model.NotificationChannel;
import
 software.amazon.awssdk.services.rekognition.model.StartPersonTrackingRequest;
import software.amazon.awssdk.services.rekognition.model.Video;
import
 software.amazon.awssdk.services.rekognition.model.StartPersonTrackingResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import
 software.amazon.awssdk.services.rekognition.model.GetPersonTrackingResponse;
import
 software.amazon.awssdk.services.rekognition.model.GetPersonTrackingRequest;
import software.amazon.awssdk.services.rekognition.model.VideoMetadata;
import software.amazon.awssdk.services.rekognition.model.PersonDetection;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class VideoPersonDetection {
    private static String startJobId = "";

    public static void main(String[] args) {
```

```
        final String usage = """

                Usage:    <bucket> <video> <topicArn> <roleArn>

                Where:
                   bucket - The name of the bucket in which the video is located
(for example, (for example, myBucket).\s
                   video - The name of video (for example, people.mp4).\s
                   topicArn - The ARN of the Amazon Simple Notification Service
(Amazon SNS) topic.\s
                   roleArn - The ARN of the AWS Identity and Access Management
(IAM) role to use.\s
                """;

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

        String bucket = args[0];
        String video = args[1];
        String topicArn = args[2];
        String roleArn = args[3];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        NotificationChannel channel = NotificationChannel.builder()
                .snsTopicArn(topicArn)
                .roleArn(roleArn)
                .build();

        startPersonLabels(rekClient, channel, bucket, video);
        getPersonDetectionResults(rekClient);
        System.out.println("This example is done!");
        rekClient.close();
    }

    public static void startPersonLabels(RekognitionClient rekClient,
            NotificationChannel channel,
            String bucket,
            String video) {
        try {
```

```
            S3Object s3Obj = S3Object.builder()
                    .bucket(bucket)
                    .name(video)
                    .build();

            Video vidOb = Video.builder()
                    .s3Object(s3Obj)
                    .build();

            StartPersonTrackingRequest personTrackingRequest =
StartPersonTrackingRequest.builder()
                    .jobTag("DetectingLabels")
                    .video(vidOb)
                    .notificationChannel(channel)
                    .build();

            StartPersonTrackingResponse labelDetectionResponse =
rekClient.startPersonTracking(personTrackingRequest);
            startJobId = labelDetectionResponse.jobId();

        } catch (RekognitionException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void getPersonDetectionResults(RekognitionClient rekClient) {
        try {
            String paginationToken = null;
            GetPersonTrackingResponse personTrackingResult = null;
            boolean finished = false;
            String status;
            int yy = 0;

            do {
                if (personTrackingResult != null)
                    paginationToken = personTrackingResult.nextToken();

                GetPersonTrackingRequest recognitionRequest =
GetPersonTrackingRequest.builder()
                        .jobId(startJobId)
                        .nextToken(paginationToken)
                        .maxResults(10)
                        .build();
```

```
                // Wait until the job succeeds
                while (!finished) {

                    personTrackingResult =
rekClient.getPersonTracking(recognitionRequest);
                    status = personTrackingResult.jobStatusAsString();

                    if (status.compareTo("SUCCEEDED") == 0)
                        finished = true;
                    else {
                        System.out.println(yy + " status is: " + status);
                        Thread.sleep(1000);
                    }
                    yy++;
                }

                finished = false;

                // Proceed when the job is done - otherwise VideoMetadata is
null.
                VideoMetadata videoMetaData =
personTrackingResult.videoMetadata();

                System.out.println("Format: " + videoMetaData.format());
                System.out.println("Codec: " + videoMetaData.codec());
                System.out.println("Duration: " +
videoMetaData.durationMillis());
                System.out.println("FrameRate: " + videoMetaData.frameRate());
                System.out.println("Job");

                List<PersonDetection> detectedPersons =
personTrackingResult.persons();
                for (PersonDetection detectedPerson : detectedPersons) {
                    long seconds = detectedPerson.timestamp() / 1000;
                    System.out.print("Sec: " + seconds + " ");
                    System.out.println("Person Identifier: " +
detectedPerson.person().index());
                    System.out.println();
                }

            } while (personTrackingResult != null &&
personTrackingResult.nextToken() != null);
```

```
        } catch (RekognitionException | InterruptedException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

Python

```
#Copyright 2018 Amazon.com, Inc. or its affiliates. All Rights Reserved.
#PDX-License-Identifier: MIT-0 (For details, see https://github.com/awsdocs/
amazon-rekognition-developer-guide/blob/master/LICENSE-SAMPLECODE.)

    # ============== People pathing ===============
    def StartPersonPathing(self):
        response=self.rek.start_person_tracking(Video={'S3Object': {'Bucket':
 self.bucket, 'Name': self.video}},
            NotificationChannel={'RoleArn': self.roleArn, 'SNSTopicArn':
 self.snsTopicArn})

        self.startJobId=response['JobId']
        print('Start Job Id: ' + self.startJobId)

    def GetPersonPathingResults(self):
        maxResults = 10
        paginationToken = ''
        finished = False

        while finished == False:
            response = self.rek.get_person_tracking(JobId=self.startJobId,
                                        MaxResults=maxResults,
                                        NextToken=paginationToken)

            print('Codec: ' + response['VideoMetadata']['Codec'])
            print('Duration: ' + str(response['VideoMetadata']
['DurationMillis']))
            print('Format: ' + response['VideoMetadata']['Format'])
            print('Frame rate: ' + str(response['VideoMetadata']['FrameRate']))
            print()

            for personDetection in response['Persons']:
                print('Index: ' + str(personDetection['Person']['Index']))
                print('Timestamp: ' + str(personDetection['Timestamp']))
```

```
                print()

            if 'NextToken' in response:
                paginationToken = response['NextToken']
            else:
                finished = True
```

En la función `main`, reemplace las líneas:

```
    analyzer.StartLabelDetection()
    if analyzer.GetSQSMessageSuccess()==True:
        analyzer.GetLabelDetectionResults()
```

por:

```
    analyzer.StartPersonPathing()
    if analyzer.GetSQSMessageSuccess()==True:
        analyzer.GetPersonPathingResults()
```

CLI

Ejecute el siguiente comando de AWS CLI para iniciar los recorridos de las personas en un vídeo.

```
aws rekognition start-person-tracking --video "{"S3Object":{"Bucket":"amzn-s3-
demo-bucket","Name":"video-name"}}" \
--notification-channel "{"SNSTopicArn":"topic-ARN","RoleArn":"role-ARN"}" \
--region region-name --profile profile-name
```

Actualice los siguientes valores:

- Cambie `amzn-s3-demo-bucket` y `video-name` por el nombre del bucket de Amazon S3 y el nombre de archivo que especificó en el paso 2.
- Cambie `region-name` por la región de AWS que está utilizando.
- Sustituya el valor de `profile-name` en la línea que crea la sesión de Rekognition por el nombre de su perfil de desarrollador.
- Reemplace `topic-ARN` por el ARN del tema de Amazon SNS que creó en el paso 3 de Configuración de Amazon Rekognition Video.

- Cambie `role-ARN` por el ARN del rol de servicio de IAM que creó en el paso 7 de Configuración de Amazon Rekognition Video.

Si accede a la CLI en un dispositivo Windows, utilice comillas dobles en lugar de comillas simples y evite las comillas dobles internas con una barra invertida (es decir, \) para corregir cualquier error del analizador que pueda encontrar. Consulte a continuación un ejemplo:

```
aws rekognition start-person-tracking --video "{\"S3Object\":{\"Bucket\":\"amzn-
s3-demo-bucket\",\"Name\":\"video-name\"}}"
--notification-channel "{\"SNSTopicArn\":\"topic-ARN\",\"RoleArn\":\"role-ARN
\"}" \
--region region-name --profile profile-name
```

Tras ejecutar el ejemplo de código de procedimiento, copie el `jobID` devuelto y envíelo al siguiente comando `GetPersonTracking` para obtener los resultados, sustituyendo `job-id-number` por el `jobID` que recibió anteriormente:

```
aws rekognition get-person-tracking --job-id job-id-number
```

> **Note**
>
> Si ya ha ejecutado un ejemplo de vídeo distinto de Análisis de un vídeo almacenado en un bucket de Amazon S3 con Java o Python (SDK), el código que se va a reemplazar podría ser diferente.

3. Ejecute el código. Los identificadores únicos de las personas de las que se realiza un seguimiento se muestran junto con el tiempo, en segundos, en que se realizó el seguimiento de las personas.

Respuesta de la operación GetPersonTracking

`GetPersonTracking` devuelve una matriz, `Persons`, de objetos PersonDetection que contienen detalles acerca de las personas detectadas en el vídeo y cuándo se ha hecho seguimiento de sus recorridos.

Puede ordenar `Persons` utilizando el parámetro de entrada `SortBy`. Especifique `TIMESTAMP` para ordenar los elementos según la hora de seguimiento del recorrido de las personas en el vídeo. Especifique `INDEX` para ordenar por personas de las que se hace seguimiento en el vídeo. Dentro de cada conjunto de resultados para una persona, los elementos se ordenan por confianza en sentido descendente según la precisión del seguimiento del recorrido. De forma predeterminada, `Persons` se devuelve ordenado por `TIMESTAMP`. El siguiente ejemplo es la respuesta JSON de `GetPersonDetection`. Los resultados se ordenan por tiempo, en milisegundos desde el inicio del vídeo, durante el que se realiza un seguimiento de los recorridos de las personas en el vídeo. En la respuesta, tenga en cuenta lo siguiente:

- Información de persona: el elemento de matriz `PersonDetection` contiene información acerca de la persona detectada. Por ejemplo, la hora a la que se detectó la persona (`Timestamp`), la posición de la persona en el fotograma de vídeo en el momento en que se detectó (`BoundingBox`) y la confianza que tiene Amazon Rekognition Video de que la persona se ha detectado correctamente (`Confidence`).

  Los rasgos faciales no se devuelven en cada marca temporal en la que se realiza el seguimiento del recorrido de la persona. Además, en algunos casos, el cuerpo de la persona a la que se hace seguimiento podría no ser visible, en cuyo caso solo se devuelve la ubicación del rostro.
- Información de paginación: el ejemplo muestra una página de información de detección de persona. Puede especificar la cantidad de elementos de persona que se van a devolver en el parámetro de entrada `MaxResults` para `GetPersonTracking`. Si existen más resultados que `MaxResults`, `GetPersonTracking` devuelve un token (`NextToken`) que se utiliza para obtener la siguiente página de resultados. Para obtener más información, consulte Obtención de los resultados del análisis de Amazon Rekognition Video.
- Índice: un identificador único para identificar la persona a lo largo del vídeo.
- Información de vídeo: la respuesta incluye información acerca del formato de vídeo (`VideoMetadata`) en cada página de información devuelta por `GetPersonDetection`.

```
{
```

```
    "JobStatus": "SUCCEEDED",
    "NextToken": "AcDymG0fSSoaI6+BBYpka5wVlqttysSPP8VvWcujMDluj1QpFo/vf
+mrMoqBGk8eUEiFlllR6g==",
    "Persons": [
        {
            "Person": {
                "BoundingBox": {
                    "Height": 0.8787037134170532,
                    "Left": 0.00572916679084301,
                    "Top": 0.12129629403352737,
                    "Width": 0.21666666865348816
                },
                "Face": {
                    "BoundingBox": {
                        "Height": 0.20000000298023224,
                        "Left": 0.029999999329447746,
                        "Top": 0.2199999988079071,
                        "Width": 0.11249999701976776
                    },
                    "Confidence": 99.85971069335938,
                    "Landmarks": [
                        {
                            "Type": "eyeLeft",
                            "X": 0.06842322647571564,
                            "Y": 0.3010137975215912
                        },
                        {
                            "Type": "eyeRight",
                            "X": 0.10543643683195114,
                            "Y": 0.29697132110595703
                        },
                        {
                            "Type": "nose",
                            "X": 0.09569807350635529,
                            "Y": 0.33701086044311523
                        },
                        {
                            "Type": "mouthLeft",
                            "X": 0.0732642263174057,
                            "Y": 0.3757539987564087
                        },
                        {
                            "Type": "mouthRight",
                            "X": 0.10589495301246643,
```

```
                        "Y": 0.3722417950630188
                    }
                ],
                "Pose": {
                    "Pitch": -0.5589138865470886,
                    "Roll": -5.1093974113464355,
                    "Yaw": 18.69594955444336
                },
                "Quality": {
                    "Brightness": 43.052337646484375,
                    "Sharpness": 99.68138885498047
                }
            },
            "Index": 0
        },
        "Timestamp": 0
    },
    {
        "Person": {
            "BoundingBox": {
                "Height": 0.9074074029922485,
                "Left": 0.24791666865348816,
                "Top": 0.09259258955717087,
                "Width": 0.375
            },
            "Face": {
                "BoundingBox": {
                    "Height": 0.23000000417232513,
                    "Left": 0.42500001192092896,
                    "Top": 0.16333332657814026,
                    "Width": 0.12937499582767487
                },
                "Confidence": 99.97504425048828,
                "Landmarks": [
                    {
                        "Type": "eyeLeft",
                        "X": 0.46415066719055176,
                        "Y": 0.2572723925113678
                    },
                    {
                        "Type": "eyeRight",
                        "X": 0.5068183541297913,
                        "Y": 0.23705792427062988
                    },
```

```
                        {
                            "Type": "nose",
                            "X": 0.49765899777412415,
                            "Y": 0.28383663296699524
                        },
                        {
                            "Type": "mouthLeft",
                            "X": 0.487221896648407,
                            "Y": 0.3452930748462677
                        },
                        {
                            "Type": "mouthRight",
                            "X": 0.5142884850502014,
                            "Y": 0.33167609572410583
                        }
                    ],
                    "Pose": {
                        "Pitch": 15.966927528381348,
                        "Roll": -15.547388076782227,
                        "Yaw": 11.34195613861084
                    },
                    "Quality": {
                        "Brightness": 44.80223083496094,
                        "Sharpness": 99.95819854736328
                    }
                },
                "Index": 1
            },
            "Timestamp": 0
        }.....

    ],
    "VideoMetadata": {
        "Codec": "h264",
        "DurationMillis": 67301,
        "FileExtension": "mp4",
        "Format": "QuickTime / MOV",
        "FrameHeight": 1080,
        "FrameRate": 29.970029830932617,
        "FrameWidth": 1920
    }
}
```

Detección de equipos de protección individual

Amazon Rekognition puede detectar el equipo de protección individual (EPI) que llevan las personas en una imagen. Puede utilizar esta información para mejorar las prácticas de seguridad en el lugar de trabajo. Por ejemplo, puede usar la detección de EPI para ayudar a determinar si los trabajadores de una obra de construcción llevan cubierta la cabeza o si los trabajadores médicos usan cubiertas para la cara y las manos. La siguiente imagen muestra algunos de los tipos de EPI que se pueden detectar.



Para detectar el PPE en una imagen, debe llamar a la DetectProtectiveEquipmentAPI y pasar una imagen de entrada. La respuesta es una estructura JSON que incluye lo siguiente.

- Las personas detectadas en la imagen.
- Las partes del cuerpo en las que se lleva puesto el EPI (cara, cabeza, mano izquierda y mano derecha).

- Los tipos de EPI que se detectan en las partes del cuerpo (cubierta facial, cubierta para manos y cubierta para la cabeza).
- En el caso de los elementos de EPI detectados, un indicador de si el EPI cubre o no la parte del cuerpo correspondiente.

Los recuadros delimitadores indican la ubicación de las personas y los objetos de EPI detectados en la imagen.

Si lo desea, puede solicitar un resumen de los objetos de EPI y las personas detectados en una imagen. Para obtener más información, consulte Descripción de EPI detectado en una imagen.

> **Note**
>
> La detección de EPI de Amazon Rekognition no realiza reconocimiento facial ni comparación facial y no puede identificar a las personas detectadas.

Tipos de EPI

DetectProtectiveEquipmentdetecta los siguientes tipos de EPP. Si desea detectar otros tipos de EPI en las imágenes, considere la posibilidad de utilizar las Etiquetas personalizadas de Amazon Rekognition para entrenar un modelo personalizado. Para obtener más información, consulte Etiquetas personalizadas de Amazon Rekognition.

Cubierta facial

`DetectProtectiveEquipment` puede detectar mascarillas comunes, como las quirúrgicas, las N95 y las mascarillas de tela.

Cubierta para la mano

`DetectProtectiveEquipment` puede detectar cubiertas para la mano, como guantes quirúrgicos y guantes de seguridad.

Cubierta para la cabeza

`DetectProtectiveEquipment` puede detectar gorros y cascos.

La API indica que se ha detectado una cubierta para la cabeza, la mano o la cara en una imagen. La API no devuelve información sobre el tipo de cubierta específica. Por ejemplo, “guante quirúrgico” para el tipo de cubierta de mano.

Confianza de detección de EPI

Amazon Rekognition realiza una predicción sobre la presencia de EPI, personas y partes del cuerpo en una imagen. La API proporciona una puntuación (50 a 100) que indica la confianza de Amazon Rekognition en la precisión de una predicción.

> **Note**
>
> Si piensa utilizar la operación `DetectProtectiveEquipment` para tomar una decisión que afecte a los derechos, la privacidad o el acceso de una persona a los servicios, le recomendamos que transmita el resultado a una persona para que lo revise y lo valide antes de tomar medidas.

Descripción de EPI detectado en una imagen

Si lo desea, puede solicitar un resumen de los objetos de EPI y las personas detectados en una imagen. Puede especificar una lista de EPI necesario (cubierta facial, cubierta para las manos o cubierta para la cabeza) y un umbral de confianza mínimo (por ejemplo, el 80 %). La respuesta incluye un resumen consolidado del identificador (ID) por imagen de las personas que llevan el EPI requerido, las personas que no tienen el EPI requerido y las personas sobre las que no se pudo tomar una decisión.

El resumen le permite responder rápidamente a preguntas como: ¿Cuántas personas no se cubren la cara? o ¿Todas las personas usan EPI? Cada persona detectada en el resumen tiene un ID único. Puede utilizar el ID para obtener información como la ubicación del recuadro delimitador de una persona que no lleve puesto el EPI.

> **Note**
>
> El ID se genera aleatoriamente a partir de un análisis por imagen y no es coherente en todas las imágenes o en varios análisis de la misma imagen.

Puede resumir las cubiertas faciales, las cubiertas para la cabeza, las cubiertas para las manos o una combinación de lo que prefiera. Para especificar los tipos de EPI necesarios, consulte Especificar los requisitos de resumen. También puede especificar un nivel de confianza mínimo (50-100) que debe cumplirse para que las detecciones se incluyan en el resumen.

Para obtener más información sobre la respuesta de resumen de `DetectProtectiveEquipment`, consulte Entender la DetectProtectiveEquipment respuesta.

Tutorial: Creación de una AWS Lambda función que detecte imágenes con PPE

Puede crear una AWS Lambda función que detecte el equipo de protección personal (EPP) en las imágenes ubicadas en un bucket de Amazon S3. Consulte el GitHub repositorio de ejemplos del SDK de AWS documentación para ver este tutorial sobre Java V2.

Comprensión de la API de detección de equipos de protección individual

La siguiente información describe la DetectProtectiveEquipmentAPI. Para ver el código de ejemplo, consulte Detección de equipos de protección individual en una imagen.

Suministrar una imagen

Puede proporcionar una imagen de entrada (formato JPG o PNG) como bytes de imagen o referencia a una imagen almacenada en un bucket de Amazon S3.

Recomendamos utilizar imágenes en las que el rostro de la persona mire hacia la cámara.

Si la imagen de entrada no está girada en una orientación de 0 grados, le recomendamos girarla a una orientación de 0 grados antes de enviarla a `DetectProtectiveEquipment`. Las imágenes en formato JPG pueden contener información de orientación en los metadatos del formato de archivo de imagen intercambiable (Exif). Puede utilizar esta información para escribir código que haga una rotación. Para obtener más información, consulte Exif Version 2.32. Las imágenes en formato PNG no contienen información sobre la orientación de la imagen.

Para transferir una imagen desde un bucket de Amazon S3, utilice un usuario con al menos los ReadOnlyAccess privilegios de Amazon S3. Utilice un usuario con privilegios de `AmazonRekognitionFullAccess` para llamar a `DetectProtectiveEquipment`.

En el siguiente ejemplo, si introduce JSON, la imagen se pasa a un bucket de Amazon S3. Para obtener más información, consulte Trabajar con imágenes. En el ejemplo se solicita un resumen de todos los tipos de EPI (cubierta para la cabeza, cubierta para las manos y cubierta para la cara) con una confianza de detección mínima (`MinConfidence`) del 80 %. Debe especificar un valor de `MinConfidence` que esté entre el 50 y el 100 %, ya que `DetectProtectiveEquipment` solo devuelve predicciones cuando la confianza de detección está entre el 50 y el 100 %. Si especifica un valor inferior al 50 %, los resultados son los mismos, especificando un valor del 50 %. Para obtener más información, consulte Especificar los requisitos de resumen.

```
{
    "Image": {
        "S3Object": {
            "Bucket": "amzn-s3-demo-bucket",
            "Name": "worker.jpg"
        }
    },
    "SummarizationAttributes": {
        "MinConfidence": 80,
        "RequiredEquipmentTypes": [
            "FACE_COVER",
            "HAND_COVER",
            "HEAD_COVER"
        ]
    }
}
```

Si tiene una amplia gama de imágenes para procesar, considere la posibilidad de utilizar AWS Batch para procesar las llamadas a `DetectProtectiveEquipment` en lotes en segundo plano.

Especificar los requisitos de resumen

Si lo desea, puede utilizar el parámetro de entrada `SummarizationAttributes` (ProtectiveEquipmentSummarizationAttributes) para solicitar información resumida sobre los tipos de PPE detectados en una imagen.

Para especificar los tipos de EPI que se van a resumir, utilice el campo de matriz de `RequiredEquipmentTypes`. En la matriz, incluya uno o más `FACE_COVER`, `HAND_COVER` o `HEAD_COVER`.

Utilice el campo `MinConfidence` para especificar una confianza de detección mínima (50-100). El resumen no incluye las personas, las partes del cuerpo, la cobertura de partes del cuerpo ni los elementos de EPI detectados con una confianza inferior a `MinConfidence`.

Para obtener información sobre la respuesta de resumen de `DetectProtectiveEquipment`, consulte Entender la DetectProtectiveEquipment respuesta.

Entender la DetectProtectiveEquipment respuesta

`DetectProtectiveEquipment` devuelve una matriz de personas detectadas en la imagen de entrada. Para cada persona, se devuelve información sobre las partes del cuerpo detectadas y los elementos de EPI. El código JSON de la siguiente imagen de un trabajador con la cabeza, las manos y la cara cubiertas es el siguiente.



En el JSON, tenga en cuenta lo siguiente.

- Personas detectadas: `Persons` es un conjunto de personas detectadas en la imagen (incluidas las personas que no llevan EPI). `DetectProtectiveEquipment` puede detectar el EPI en hasta 15 personas detectadas en una imagen. Cada ProtectiveEquipmentPersonobjeto de la matriz contiene un carné de identidad, un recuadro delimitador para la persona, las partes del cuerpo detectadas y los elementos de protección personal detectados. El valor `Confidence` en `ProtectiveEquipmentPerson` indica el porcentaje de confianza que Amazon Rekognition tiene en que el recuadro delimitador contiene una persona.
- Partes del cuerpo: `BodyParts` es un conjunto de partes del cuerpo (ProtectiveEquipmentBodyPart) detectadas en una persona (incluidas las partes del cuerpo que no están cubiertas por el EPP). Cada `ProtectiveEquipmentBodyPart` incluye el nombre (`Name`) de la parte del cuerpo detectada. `DetectProtectEquipment` puede detectar partes del cuerpo de la cara, la cabeza y las manos. El campo `Confidence` en `ProtectiveEquipmentBodyPart` indica el porcentaje de confianza que Amazon Rekognition tiene en la precisión de detección de la parte del cuerpo.
- Elementos de EPI: la matriz de `EquipmentDetections` en un objeto `ProtectiveEquipmentBodyPart` contiene una serie de elementos de EPI detectados. Cada EquipmentDetectionobjeto contiene los siguientes campos.
  - `Type`: el tipo de EPI detectado.
  - `BoundingBox`: un recuadro delimitador alrededor del EPI detectado.
  - `Confidence`: la confianza que Amazon Rekognition tiene de que el recuadro delimitador contiene el EPI detectado.
  - `CoversBodyPart`: indica si el EPI detectado se encuentra en la parte del cuerpo correspondiente.

  El CoversBodyPartcampo `Value` es un valor booleano que indica si el EPP detectado se encuentra en la parte del cuerpo correspondiente. El campo `Confidence` indica la confianza en la predicción. Se puede utilizar `CoversBodyPart` para filtrar los casos en los que el EPI detectado aparezca en la imagen, pero no en la persona.

> **Note**
>
> `CoversBodyPart` no indica ni implica que la persona esté adecuadamente protegida por el EPI o que el equipo de protección en sí esté debidamente colocado.

- Información resumida: `Summary` contiene la información resumida especificada en el parámetro de entrada `SummarizationAttributes`. Para obtener más información, consulte Especificar los requisitos de resumen.

  `Summary`es un objeto de tipo ProtectiveEquipmentSummaryque contiene la siguiente información.

  - `PersonsWithRequiredEquipment`— Un conjunto IDs de personas en el que cada persona cumple los siguientes criterios.
    - La persona lleva puesto todo el EPI especificado en el parámetro de entrada `SummarizationAttributes`.
    - El `Confidence` para la persona (`ProtectiveEquipmentPerson`), la parte del cuerpo (`ProtectiveEquipmentBodyPart`) y el equipo de protección (`EquipmentDetection`) es igual o superior al umbral de confianza mínimo especificado (`MinConfidence`).
    - El valor de `CoversBodyPart` para todos los elementos del EPI es true.
  - `PersonsWithoutRequiredEquipment`— Un conjunto IDs de personas que cumplen uno de los siguientes criterios.
    - El valor `Confidence` de la persona (`ProtectiveEquipmentPerson`), la parte del cuerpo (`ProtectiveEquipmentBodyPart`) y la cobertura de la parte del cuerpo (`CoversBodyPart`) superan el umbral de confianza mínimo especificado (`MinConfidence`), pero a la persona le faltan uno o más EPI específicos (`SummarizationAttributes`).
    - El valor de `CoversBodyPart` es falso para cualquier EPI (`SummarizationAttributes`) especificado que tenga un valor de `Confidence` superior al umbral de confianza mínimo especificado (`MinConfidence`). La persona también tiene todo el EPI especificado (`SummarizationAttributes`) y los valores `Confidence` de persona (`ProtectiveEquipmentPerson`), parte del cuerpo (`ProtectiveEquipmentBodyPart`) y equipo de protección (`EquipmentDetection`) son superiores o iguales al umbral mínimo de confianza (`MinConfidence`).
  - `PersonsIndeterminate`— Un conjunto IDs de personas detectadas en el que el `Confidence` valor correspondiente a la persona (`ProtectiveEquipmentPerson`), la parte del cuerpo (`ProtectiveEquipmentBodyPart`), el equipo de protección (`EquipmentDetection`) o el valor `CoversBodyPart` booleano es inferior al umbral mínimo de confianza especificado (`MinConfidence`).

  Utilice el tamaño de la matriz para obtener el recuento de un resumen concreto. Por ejemplo, el valor de `PersonsWithRequiredEquipment` indica el número de personas detectadas que llevan el tipo de EPI especificado.

Puede usar el ID de la persona para obtener más información sobre una persona, como la ubicación del recuadro delimitador de la persona. El ID de la persona se asigna al campo de ID de un objeto `ProtectiveEquipmentPerson` devuelto en `Persons` (matriz de `ProtectiveEquipmentPerson`). A continuación, puede obtener el cuadro delimitador y otra información del objeto `ProtectiveEquipmentPerson` correspondiente.


```
{
   "ProtectiveEquipmentModelVersion": "1.0",
   "Persons": [
       {
           "BodyParts": [
               {
                   "Name": "FACE",
                   "Confidence": 99.99861145019531,
                   "EquipmentDetections": [
                       {
                           "BoundingBox": {
                               "Width": 0.14528800547122955,
                               "Height": 0.14956723153591156,
                               "Left": 0.4363413453102112,
                               "Top": 0.34203192591667175
                           },
                           "Confidence": 99.90001678466797,
                           "Type": "FACE_COVER",
                           "CoversBodyPart": {
                               "Confidence": 98.0676498413086,
                               "Value": true
                           }
                       }
                   ]
               },
               {
                   "Name": "LEFT_HAND",
                   "Confidence": 96.9786376953125,
                   "EquipmentDetections": [
                       {
                           "BoundingBox": {
                               "Width": 0.14495663344860077,
                               "Height": 0.12936046719551086,
                               "Left": 0.5114737153053284,
```

```
                        "Top": 0.5744519829750061
                    },
                    "Confidence": 83.72270965576172,
                    "Type": "HAND_COVER",
                    "CoversBodyPart": {
                        "Confidence": 96.9288558959961,
                        "Value": true
                    }
                }
            ]
        },
        {
            "Name": "RIGHT_HAND",
            "Confidence": 99.82939147949219,
            "EquipmentDetections": [
                {
                    "BoundingBox": {
                        "Width": 0.20971858501434326,
                        "Height": 0.20528452098369598,
                        "Left": 0.2711356580257416,
                        "Top": 0.6750612258911133
                    },
                    "Confidence": 95.70789337158203,
                    "Type": "HAND_COVER",
                    "CoversBodyPart": {
                        "Confidence": 99.85433197021484,
                        "Value": true
                    }
                }
            ]
        },
        {
            "Name": "HEAD",
            "Confidence": 99.9999008178711,
            "EquipmentDetections": [
                {
                    "BoundingBox": {
                        "Width": 0.24350935220718384,
                        "Height": 0.34623199701309204,
                        "Left": 0.43011072278022766,
                        "Top": 0.01103297434747219
                    },
                    "Confidence": 83.88762664794922,
                    "Type": "HEAD_COVER",
```

```
                            "CoversBodyPart": {
                                "Confidence": 99.96485900878906,
                                "Value": true
                            }
                        }
                    ]
                }
            ],
            "BoundingBox": {
                "Width": 0.7403100728988647,
                "Height": 0.9412225484848022,
                "Left": 0.02214839495718479,
                "Top": 0.03134796395897865
            },
            "Confidence": 99.98855590820312,
            "Id": 0
        }
    ],
    "Summary": {
        "PersonsWithRequiredEquipment": [
            0
        ],
        "PersonsWithoutRequiredEquipment": [],
        "PersonsIndeterminate": []
    }
}
```


Detección de equipos de protección individual en una imagen

Para detectar el equipo de protección personal (EPP) en las personas de una imagen, utilice la operación API DetectProtectiveEquipmentsin almacenamiento.

Puede proporcionar la imagen de entrada como matriz de bytes de imagen (bytes de imagen con codificación base64) o como un objeto de Amazon S3, utilizando el AWS SDK o la AWS Command Line Interface (AWS CLI). En estos ejemplos, se utiliza una imagen almacenada en un bucket de Amazon S3. Para obtener más información, consulte Trabajar con imágenes.

Para detectar el EPI en las personas de una imagen

1. Si aún no lo ha hecho:

a. Cree o actualice un usuario con los permisos `AmazonRekognitionFullAccess` y `AmazonS3ReadOnlyAccess`. Para obtener más información, consulte Paso 1: Configurar una cuenta de AWS y crear un usuario.

b. Instale y configure el AWS CLI y el AWS SDKs. Para obtener más información, consulte Paso 2: Configure y AWS CLI AWS SDKs.

2. Suba una imagen que contenga una o varias personas llevando EPI en su bucket de S3.

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

3. Consulte los siguientes ejemplos para llamar a la operación `DetectProtectiveEquipment`. Para obtener información sobre cómo mostrar los cuadros delimitadores en una imagen, consulte Visualización de cuadros delimitadores.

   Java

   En este ejemplo, se muestra información sobre los elementos de EPI detectados en las personas detectadas en una imagen.

   Cambie el valor de `amzn-s3-demo-bucket` por el nombre del bucket de Amazon S3 que contiene su imagen. Cambie el valor de `photo` por el nombre de su archivo de imagen.

```
//Copyright 2020 Amazon.com, Inc. or its affiliates. All Rights Reserved.
//PDX-License-Identifier: MIT-0 (For details, see https://github.com/awsdocs/
amazon-rekognition-developer-guide/blob/master/LICENSE-SAMPLECODE.)

package com.amazonaws.samples;
import com.amazonaws.client.builder.AwsClientBuilder;
import com.amazonaws.services.rekognition.AmazonRekognition;
import com.amazonaws.services.rekognition.AmazonRekognitionClientBuilder;
import com.amazonaws.services.rekognition.model.AmazonRekognitionException;
import com.amazonaws.services.rekognition.model.Image;
import com.amazonaws.services.rekognition.model.ProtectiveEquipmentBodyPart;
import com.amazonaws.services.rekognition.model.S3Object;
import com.amazonaws.services.rekognition.model.ProtectiveEquipmentPerson;
import
 com.amazonaws.services.rekognition.model.ProtectiveEquipmentSummarizationAttributes;

import java.util.List;
import com.amazonaws.services.rekognition.model.BoundingBox;
```

```
import
 com.amazonaws.services.rekognition.model.DetectProtectiveEquipmentRequest;
import com.amazonaws.services.rekognition.model.DetectProtectiveEquipmentResult;
import com.amazonaws.services.rekognition.model.EquipmentDetection;


public class DetectPPE {

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

        String photo = "photo";
        String bucket = "bucket";


        AmazonRekognition rekognitionClient =
 AmazonRekognitionClientBuilder.defaultClient();

        ProtectiveEquipmentSummarizationAttributes summaryAttributes = new
 ProtectiveEquipmentSummarizationAttributes()
                .withMinConfidence(80F)
                .withRequiredEquipmentTypes("FACE_COVER", "HAND_COVER",
 "HEAD_COVER");

        DetectProtectiveEquipmentRequest request = new
 DetectProtectiveEquipmentRequest()
                .withImage(new Image()
                        .withS3Object(new S3Object()
                                .withName(photo).withBucket(bucket)))
                .withSummarizationAttributes(summaryAttributes);

        try {
            System.out.println("Detected PPE for people in image " + photo);
            System.out.println("Detected people\n---------------");
            DetectProtectiveEquipmentResult result =
 rekognitionClient.detectProtectiveEquipment(request);


            List <ProtectiveEquipmentPerson> persons = result.getPersons();


            for (ProtectiveEquipmentPerson person: persons) {
                System.out.println("ID: " + person.getId());
                List<ProtectiveEquipmentBodyPart>
 bodyParts=person.getBodyParts();
```

```
                if (bodyParts.isEmpty()){
                    System.out.println("\tNo body parts detected");
                } else
                    for (ProtectiveEquipmentBodyPart bodyPart: bodyParts) {
                        System.out.println("\t" + bodyPart.getName() + ".
Confidence: " + bodyPart.getConfidence().toString());



                        List<EquipmentDetection>
equipmentDetections=bodyPart.getEquipmentDetections();

                        if (equipmentDetections.isEmpty()){
                            System.out.println("\t\tNo PPE Detected on " +
bodyPart.getName());

                        }
                        else {
                            for (EquipmentDetection item: equipmentDetections) {
                                System.out.println("\t\tItem: " + item.getType()
+ ". Confidence: " + item.getConfidence().toString());
                                System.out.println("\t\tCovers body part: "
                                        +
item.getCoversBodyPart().getValue().toString() + ". Confidence: " +
item.getCoversBodyPart().getConfidence().toString());

                                System.out.println("\t\tBounding Box");
                                BoundingBox box =item.getBoundingBox();

                                System.out.println("\t\tLeft: "
+box.getLeft().toString());
                                System.out.println("\t\tTop: " +
box.getTop().toString());
                                System.out.println("\t\tWidth: " +
box.getWidth().toString());
                                System.out.println("\t\tHeight: " +
box.getHeight().toString());
                                System.out.println("\t\tConfidence: " +
item.getConfidence().toString());
                                System.out.println();
                            }
                        }

                    }
```

```
            }
            System.out.println("Person ID Summary\n-----------------");

            //List<Integer> list=;
            DisplaySummary("With required equipment",
 result.getSummary().getPersonsWithRequiredEquipment());
            DisplaySummary("Without required equipment",
 result.getSummary().getPersonsWithoutRequiredEquipment());
            DisplaySummary("Indeterminate",
 result.getSummary().getPersonsIndeterminate());


        } catch(AmazonRekognitionException e) {
            e.printStackTrace();
        }
    }
    static void DisplaySummary(String summaryType,List<Integer> idList)
    {
        System.out.print(summaryType + "\n\tIDs  ");
        if (idList.size()==0) {
            System.out.println("None");
        }
        else {
            int count=0;
            for (Integer id: idList ) {
                if (count++ == idList.size()-1) {
                    System.out.println(id.toString());
                }
                else {
                    System.out.print(id.toString() + ", ");
                }
            }
        }

        System.out.println();


    }
}
```

Java V2

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

```
//snippet-start:[rekognition.java2.detect_ppe.import]
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.core.ResponseBytes;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.BoundingBox;
import
 software.amazon.awssdk.services.rekognition.model.DetectProtectiveEquipmentRequest;
import
 software.amazon.awssdk.services.rekognition.model.DetectProtectiveEquipmentResponse;
import software.amazon.awssdk.services.rekognition.model.EquipmentDetection;
import
 software.amazon.awssdk.services.rekognition.model.ProtectiveEquipmentBodyPart;
import
 software.amazon.awssdk.services.rekognition.model.ProtectiveEquipmentSummarizationAttri
import software.amazon.awssdk.services.rekognition.model.Image;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.GetObjectResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;
import
 software.amazon.awssdk.services.rekognition.model.ProtectiveEquipmentPerson;
import java.io.ByteArrayInputStream;
import java.io.InputStream;
import java.util.List;
//snippet-end:[rekognition.java2.detect_ppe.import]

/**
* Before running this Java V2 code example, set up your development environment,
 including your credentials.
*
* For more information, see the following documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
*/
```

```
public class DetectPPE {

 public static void main(String[] args) {

     final String usage = "\n" +
         "Usage: " +
         "   <sourceImage> <bucketName>\n\n" +
         "Where:\n" +
         "   sourceImage - The name of the image in an Amazon S3 bucket (for
 example, people.png). \n\n" +
         "   bucketName - The name of the Amazon S3 bucket (for example, amzn-
s3-demo-bucket). \n\n";

     if (args.length != 2) {
          System.out.println(usage);
          System.exit(1);
     }

     String sourceImage = args[0];
     String bucketName = args[1];
     Region region = Region.US_WEST_2;
     S3Client s3 = S3Client.builder()
         .region(region)
         .credentialsProvider(ProfileCredentialsProvider.create("profile-name"))
         .build();

     RekognitionClient rekClient = RekognitionClient.builder()
         .region(region)
         .credentialsProvider(ProfileCredentialsProvider.create("profile-name"))
         .build();

     displayGear(s3, rekClient, sourceImage, bucketName) ;
     s3.close();
     rekClient.close();
     System.out.println("This example is done!");
 }

 // snippet-start:[rekognition.java2.detect_ppe.main]
 public static void displayGear(S3Client s3,
                                RekognitionClient rekClient,
                                String sourceImage,
                                String bucketName) {

     byte[] data = getObjectBytes (s3, bucketName, sourceImage);
```

```
    InputStream is = new ByteArrayInputStream(data);

    try {
        ProtectiveEquipmentSummarizationAttributes summarizationAttributes =
ProtectiveEquipmentSummarizationAttributes.builder()
            .minConfidence(80F)
            .requiredEquipmentTypesWithStrings("FACE_COVER", "HAND_COVER",
"HEAD_COVER")
            .build();

        SdkBytes sourceBytes = SdkBytes.fromInputStream(is);
        software.amazon.awssdk.services.rekognition.model.Image souImage =
Image.builder()
            .bytes(sourceBytes)
            .build();

        DetectProtectiveEquipmentRequest request =
DetectProtectiveEquipmentRequest.builder()
            .image(souImage)
            .summarizationAttributes(summarizationAttributes)
            .build();

        DetectProtectiveEquipmentResponse result =
rekClient.detectProtectiveEquipment(request);
        List<ProtectiveEquipmentPerson> persons = result.persons();
        for (ProtectiveEquipmentPerson person: persons) {
            System.out.println("ID: " + person.id());
            List<ProtectiveEquipmentBodyPart> bodyParts=person.bodyParts();
            if (bodyParts.isEmpty()){
                System.out.println("\tNo body parts detected");
            } else
                for (ProtectiveEquipmentBodyPart bodyPart: bodyParts) {
                    System.out.println("\t" + bodyPart.name() + ". Confidence:
" + bodyPart.confidence().toString());
                    List<EquipmentDetection>
equipmentDetections=bodyPart.equipmentDetections();

                    if (equipmentDetections.isEmpty()){
                        System.out.println("\t\tNo PPE Detected on " +
bodyPart.name());
                    } else {
                        for (EquipmentDetection item: equipmentDetections) {
                            System.out.println("\t\tItem: " + item.type() + ".
Confidence: " + item.confidence().toString());
```

```
                            System.out.println("\t\tCovers body part: "
                                    + item.coversBodyPart().value().toString()
+ ". Confidence: " + item.coversBodyPart().confidence().toString());

                            System.out.println("\t\tBounding Box");
                            BoundingBox box =item.boundingBox();
                            System.out.println("\t\tLeft: "
+box.left().toString());
                            System.out.println("\t\tTop: " +
box.top().toString());
                            System.out.println("\t\tWidth: " +
box.width().toString());
                            System.out.println("\t\tHeight: " +
box.height().toString());
                            System.out.println("\t\tConfidence: " +
item.confidence().toString());
                            System.out.println();
                        }
                    }
                }
        }
        System.out.println("Person ID Summary\n-----------------");

        displaySummary("With required equipment",
result.summary().personsWithRequiredEquipment());
        displaySummary("Without required equipment",
result.summary().personsWithoutRequiredEquipment());
        displaySummary("Indeterminate",
result.summary().personsIndeterminate());

    } catch (RekognitionException e) {
        e.printStackTrace();
        System.exit(1);
    }
}

public static byte[] getObjectBytes (S3Client s3, String bucketName, String
keyName) {

    try {
        GetObjectRequest objectRequest = GetObjectRequest
            .builder()
            .key(keyName)
            .bucket(bucketName)
```

```
                .build();

        ResponseBytes<GetObjectResponse> objectBytes =
 s3.getObjectAsBytes(objectRequest);
        return objectBytes.asByteArray();

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return null;
 }

 static void displaySummary(String summaryType,List<Integer> idList) {
    System.out.print(summaryType + "\n\tIDs  ");
    if (idList.size()==0) {
        System.out.println("None");
    } else {
        int count=0;
        for (Integer id: idList ) {
            if (count++ == idList.size()-1) {
                System.out.println(id.toString());
            } else {
                System.out.print(id.toString() + ", ");
            }
        }
    }
    System.out.println();
 }
 // snippet-end:[rekognition.java2.detect_ppe.main]
}
```

AWS CLI

Este AWS CLI comando solicita un resumen del PPE y muestra el resultado JSON de la operación `detect-protective-equipment` CLI.

Cambie `amzn-s3-demo-bucket` por el nombre del bucket de Amazon S3 que contiene una imagen. Cambie el valor de `input.jpg` por el nombre de la imagen que desee usar.

```
aws rekognition detect-protective-equipment \
  --image "S3Object={Bucket=amzn-s3-demo-bucket,Name=input.jpg}" \
```

```
  --summarization-attributes
 "MinConfidence=80,RequiredEquipmentTypes=['FACE_COVER','HAND_COVER','HEAD_COVER']"
```

Este AWS CLI comando muestra el resultado JSON de la operación `detect-protective-equipment` CLI.

Cambie `amzn-s3-demo-bucket` por el nombre del bucket de Amazon S3 que contiene una imagen. Cambie el valor de `input.jpg` por el nombre de la imagen que desee usar.

```
aws rekognition detect-protective-equipment \
  --image "S3Object={Bucket=amzn-s3-demo-bucket,Name=input.jpg}"
```

Python

En este ejemplo, se muestra información sobre los elementos de EPI detectados en las personas detectadas en una imagen.

Cambie el valor de `amzn-s3-demo-bucket` por el nombre del bucket de Amazon S3 que contiene su imagen. Cambie el valor de `photo` por el nombre de su archivo de imagen. Sustituya el valor de `profile_name` en la línea que crea la sesión de Rekognition por el nombre de su perfil de desarrollador.

```
# Copyright 2020 Amazon.com, Inc. or its affiliates. All Rights Reserved.
# PDX-License-Identifier: MIT-0 (For details, see https://github.com/awsdocs/
amazon-rekognition-developer-guide/blob/master/LICENSE-SAMPLECODE.)

import boto3

def detect_ppe(photo, bucket):

    session = boto3.Session(profile_name='profile-name')
    client = session.client('rekognition')

    response = client.detect_protective_equipment(Image={'S3Object': {'Bucket':
 bucket, 'Name': photo}},

 SummarizationAttributes={'MinConfidence': 80,

 'RequiredEquipmentTypes': ['FACE_COVER',

                      'HAND_COVER',
```

```
                    'HEAD_COVER']})

    print('Detected PPE for people in image ' + photo)
    print('\nDetected people\n---------------')
    for person in response['Persons']:

        print('Person ID: ' + str(person['Id']))
        print('Body Parts\n----------')
        body_parts = person['BodyParts']
        if len(body_parts) == 0:
            print('No body parts found')
        else:
            for body_part in body_parts:
                print('\t' + body_part['Name'] + '\n\t\tConfidence: ' +
 str(body_part['Confidence']))
                print('\n\t\tDetected PPE\n\t\t------------')
                ppe_items = body_part['EquipmentDetections']
                if len(ppe_items) == 0:
                    print('\t\tNo PPE detected on ' + body_part['Name'])
                else:
                    for ppe_item in ppe_items:
                        print('\t\t' + ppe_item['Type'] + '\n\t\t\tConfidence: '
 + str(ppe_item['Confidence']))
                        print('\t\tCovers body part: ' + str(
                            ppe_item['CoversBodyPart']['Value']) + '\n\t\t
\tConfidence: ' + str(
                            ppe_item['CoversBodyPart']['Confidence']))
                        print('\t\tBounding Box:')
                        print('\t\t\tTop: ' + str(ppe_item['BoundingBox']
['Top']))
                        print('\t\t\tLeft: ' + str(ppe_item['BoundingBox']
['Left']))
                        print('\t\t\tWidth: ' + str(ppe_item['BoundingBox']
['Width']))
                        print('\t\t\tHeight: ' + str(ppe_item['BoundingBox']
['Height']))
                        print('\t\t\tConfidence: ' +
 str(ppe_item['Confidence']))
            print()
        print()

    print('Person ID Summary\n----------------')
```

```
    display_summary('With required equipment', response['Summary']
['PersonsWithRequiredEquipment'])
    display_summary('Without required equipment', response['Summary']
['PersonsWithoutRequiredEquipment'])
    display_summary('Indeterminate', response['Summary']
['PersonsIndeterminate'])

    print()
    return len(response['Persons'])

# Display summary information for supplied summary.
def display_summary(summary_type, summary):
    print(summary_type + '\n\tIDs: ', end='')
    if (len(summary) == 0):
        print('None')
    else:
        for num, id in enumerate(summary, start=0):
            if num == len(summary) - 1:
                print(id)
            else:
                print(str(id) + ', ', end='')

def main():
    photo = 'photo-name'
    bucket = 'amzn-s3-demo-bucket'
    person_count = detect_ppe(photo, bucket)
    print("Persons detected: " + str(person_count))

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

Ejemplo: dibujar recuadros delimitadores alrededor de las mascarillas

En los siguientes ejemplos, se muestra cómo dibujar recuadros delimitadores alrededor de las mascarillas detectadas en personas. Para ver un ejemplo que utilice AWS Lambda Amazon DynamoDB, consulte el repositorio de ejemplos AWS del SDK de documentación. GitHub

Para detectar mascarillas, utilice la operación API que DetectProtectiveEquipmentno es de almacenamiento. La imagen se carga desde el sistema de archivos local. Puede proporcionar la

imagen de entrada a `DetectProtectiveEquipment` como una matriz de bytes de imagen (bytes con codificación en base64). Para obtener más información, consulte Trabajar con imágenes.

El ejemplo muestra un recuadro delimitador alrededor de las mascarillas detectadas. El recuadro delimitador es verde si la mascarilla cubre completamente la parte del cuerpo. De lo contrario, aparecerá un cuadro delimitador rojo. Como advertencia, si la confianza de la detección es inferior al valor de confianza especificado, aparecerá un recuadro delimitador amarillo dentro del recuadro delimitador de la mascarilla. Si no se detecta ninguna mascarilla, se dibuja un cuadro delimitador rojo alrededor de la persona.

El resultado de la imagen es similar al siguiente.



Para mostrar los recuadros delimitadores en las mascarillas detectadas

1. Si aún no lo ha hecho:

a. Cree o actualice un usuario de `AmazonRekognitionFullAccess` con permisos. Para obtener más información, consulte Paso 1: Configurar una cuenta de AWS y crear un usuario.

b. Instale y configure el AWS CLI y el AWS SDKs. Para obtener más información, consulte Paso 2: Configure y AWS CLI AWS SDKs.

2. Consulte los siguientes ejemplos para llamar a la operación `DetectProtectiveEquipment`. Para obtener información sobre cómo mostrar los cuadros delimitadores en una imagen, consulte Visualización de cuadros delimitadores.

   Java

   En la función `main`, cambie lo siguiente:

   - El valor de `photo` a la ruta y el nombre de archivo de un archivo de imagen local (PNG o JPEG).
   - El valor de `confidence` al nivel de confianza deseado (50-100).

```
//Loads images, detects faces and draws bounding boxes.Determines exif
 orientation, if necessary.
package com.amazonaws.samples;


import java.awt.*;
import java.awt.image.BufferedImage;
import java.util.List;
import javax.imageio.ImageIO;
import javax.swing.*;

import java.io.ByteArrayInputStream;
import java.io.ByteArrayOutputStream;
import java.io.File;
import java.io.FileInputStream;
import java.io.InputStream;
import java.nio.ByteBuffer;
import com.amazonaws.util.IOUtils;

import com.amazonaws.client.builder.AwsClientBuilder;
import com.amazonaws.services.rekognition.AmazonRekognition;
```

```
import com.amazonaws.services.rekognition.AmazonRekognitionClientBuilder;
import com.amazonaws.services.rekognition.model.BoundingBox;
import
 com.amazonaws.services.rekognition.model.DetectProtectiveEquipmentRequest;
import com.amazonaws.services.rekognition.model.DetectProtectiveEquipmentResult;
import com.amazonaws.services.rekognition.model.EquipmentDetection;
import com.amazonaws.services.rekognition.model.Image;
import com.amazonaws.services.rekognition.model.ProtectiveEquipmentBodyPart;
import com.amazonaws.services.rekognition.model.ProtectiveEquipmentPerson;

// Calls DetectFaces and displays a bounding box around each detected image.
public class PPEBoundingBox extends JPanel {

    private static final long serialVersionUID = 1L;

    BufferedImage image;
    static int scale;
    DetectProtectiveEquipmentResult result;
    float confidence=80;

    public PPEBoundingBox(DetectProtectiveEquipmentResult ppeResult,
 BufferedImage bufImage, float requiredConfidence) throws Exception {
        super();
        scale = 2; // increase to shrink image size.

        result = ppeResult;
        image = bufImage;

        confidence=requiredConfidence;
    }
    // Draws the bounding box around the detected faces.
    public void paintComponent(Graphics g) {
        float left = 0;
        float top = 0;
        int height = image.getHeight(this);
        int width = image.getWidth(this);
        int offset=20;

        Graphics2D g2d = (Graphics2D) g; // Create a Java2D version of g.

        // Draw the image.
        g2d.drawImage(image, 0, 0, width / scale, height / scale, this);
        g2d.setColor(new Color(0, 212, 0));
```

```
        // Iterate through detected persons and display bounding boxes.
        List<ProtectiveEquipmentPerson> persons = result.getPersons();

        for (ProtectiveEquipmentPerson person: persons) {
            BoundingBox boxPerson = person.getBoundingBox();
            left = width * boxPerson.getLeft();
            top = height * boxPerson.getTop();
            Boolean foundMask=false;

            List<ProtectiveEquipmentBodyPart> bodyParts=person.getBodyParts();

            if (bodyParts.isEmpty()==false)
             {
                    //body parts detected

                    for (ProtectiveEquipmentBodyPart bodyPart: bodyParts) {

                        List<EquipmentDetection>
equipmentDetections=bodyPart.getEquipmentDetections();

                        for (EquipmentDetection item: equipmentDetections) {

                            if (item.getType().contentEquals("FACE_COVER"))
                            {
                                // Draw green or red bounding box depending on
mask coverage.
                                foundMask=true;
                                BoundingBox box =item.getBoundingBox();
                                left = width * box.getLeft();
                                top = height * box.getTop();
                                Color maskColor=new Color( 0, 212, 0);

                                if (item.getCoversBodyPart().getValue()==false)
{
                                    // red bounding box
                                    maskColor=new Color( 255, 0, 0);
                                }
                                g2d.setColor(maskColor);
                                g2d.drawRect(Math.round(left / scale),
Math.round(top / scale),
                                        Math.round((width * box.getWidth()) /
scale), Math.round((height * box.getHeight())) / scale);

                                // Check confidence is > supplied confidence.
```

```
                                if (item.getCoversBodyPart().getConfidence()<
confidence)
                                {
                                    // Draw a yellow bounding box inside face
mask bounding box
                                    maskColor=new Color( 255, 255, 0);
                                    g2d.setColor(maskColor);
                                    g2d.drawRect(Math.round((left + offset) /
scale),
                                            Math.round((top + offset) / scale),
                                            Math.round((width *
box.getWidth())- (offset * 2 ))/ scale,
                                            Math.round((height *
box.getHeight()) -( offset* 2)) / scale);
                                }

                            }
                        }

                    }

                }

           // Didn't find a mask, so draw person bounding box red
           if (foundMask==false) {

               left = width * boxPerson.getLeft();
               top = height * boxPerson.getTop();
               g2d.setColor(new Color(255, 0, 0));
               g2d.drawRect(Math.round(left / scale), Math.round(top / scale),
                       Math.round(((width) * boxPerson.getWidth()) / scale),
Math.round((height * boxPerson.getHeight())) / scale);
           }
        }

   }


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

       String photo = "photo";

       float confidence =80;
```

```
        int height = 0;
        int width = 0;

        BufferedImage image = null;
        ByteBuffer imageBytes;

        // Get image bytes for call to DetectProtectiveEquipment
        try (InputStream inputStream = new FileInputStream(new File(photo))) {
            imageBytes = ByteBuffer.wrap(IOUtils.toByteArray(inputStream));
        }

        //Get image for display
        InputStream imageBytesStream;
        imageBytesStream = new ByteArrayInputStream(imageBytes.array());

        ByteArrayOutputStream baos = new ByteArrayOutputStream();
        image=ImageIO.read(imageBytesStream);
        ImageIO.write(image, "jpg", baos);
        width = image.getWidth();
        height = image.getHeight();

        //Get Rekognition client
        AmazonRekognition rekognitionClient =
AmazonRekognitionClientBuilder.defaultClient();


        // Call DetectProtectiveEquipment
        DetectProtectiveEquipmentRequest request = new
DetectProtectiveEquipmentRequest()
                .withImage(new Image()
                        .withBytes(imageBytes));

        DetectProtectiveEquipmentResult result =
rekognitionClient.detectProtectiveEquipment(request);


        // Create frame and panel.
        JFrame frame = new JFrame("Detect PPE");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        PPEBoundingBox panel = new PPEBoundingBox(result, image, confidence);
        panel.setPreferredSize(new Dimension(image.getWidth() / scale,
image.getHeight() / scale));
        frame.setContentPane(panel);
```

```
        frame.pack();
        frame.setVisible(true);

    }
}
```

Java V2

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

```
import java.awt.*;
import java.awt.image.BufferedImage;
import java.io.*;
import java.util.List;
import javax.imageio.ImageIO;
import javax.swing.*;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.core.ResponseBytes;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.model.BoundingBox;
import
 software.amazon.awssdk.services.rekognition.model.DetectProtectiveEquipmentRequest;
import software.amazon.awssdk.services.rekognition.model.EquipmentDetection;
import
 software.amazon.awssdk.services.rekognition.model.ProtectiveEquipmentBodyPart;
import
 software.amazon.awssdk.services.rekognition.model.ProtectiveEquipmentPerson;
import
 software.amazon.awssdk.services.rekognition.model.ProtectiveEquipmentSummarizationAttri
import software.amazon.awssdk.services.rekognition.model.Image;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.GetObjectResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;
import
 software.amazon.awssdk.services.rekognition.model.DetectProtectiveEquipmentResponse;
//snippet-end:[rekognition.java2.display_mask.import]
```

```
/**
* Before running this Java V2 code example, set up your development environment,
 including your credentials.
*
* For more information, see the following documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
*/
public class PPEBoundingBoxFrame extends JPanel {

 DetectProtectiveEquipmentResponse result;
 static BufferedImage image;
 static int scale;
 float confidence;

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

     final String usage = "\n" +
         "Usage: " +
         "   <sourceImage> <bucketName>\n\n" +
         "Where:\n" +
         "   sourceImage - The name of the image in an Amazon S3 bucket that
 shows a person wearing a mask (for example, masks.png). \n\n" +
         "   bucketName - The name of the Amazon S3 bucket (for example, amzn-
s3-demo-bucket). \n\n";

     if (args.length != 2) {
         System.out.println(usage);
         System.exit(1);
     }

     String sourceImage = args[0];
     String bucketName = args[1];
     Region region = Region.US_EAST_1;
     S3Client s3 = S3Client.builder()
         .region(region)
         .credentialsProvider(ProfileCredentialsProvider.create("profile-name"))
         .build();

     RekognitionClient rekClient = RekognitionClient.builder()
         .region(region)
         .credentialsProvider(ProfileCredentialsProvider.create("profile-name"))
         .build();
```

```
        displayGear(s3, rekClient, sourceImage, bucketName);
        s3.close();
        rekClient.close();
    }

    // snippet-start:[rekognition.java2.display_mask.main]
    public static void displayGear(S3Client s3,
                                   RekognitionClient rekClient,
                                   String sourceImage,
                                   String bucketName) {
       float confidence = 80;
       byte[] data = getObjectBytes(s3, bucketName, sourceImage);
       InputStream is = new ByteArrayInputStream(data);

       try {
           ProtectiveEquipmentSummarizationAttributes summarizationAttributes =
    ProtectiveEquipmentSummarizationAttributes.builder()
                .minConfidence(70F)
                .requiredEquipmentTypesWithStrings("FACE_COVER")
                .build();

           SdkBytes sourceBytes = SdkBytes.fromInputStream(is);
           image = ImageIO.read(sourceBytes.asInputStream());

           // Create an Image object for the source image.
           software.amazon.awssdk.services.rekognition.model.Image souImage =
    Image.builder()
                .bytes(sourceBytes)
                .build();

           DetectProtectiveEquipmentRequest request =
    DetectProtectiveEquipmentRequest.builder()
                .image(souImage)
                .summarizationAttributes(summarizationAttributes)
                .build();

           DetectProtectiveEquipmentResponse result =
    rekClient.detectProtectiveEquipment(request);
           JFrame frame = new JFrame("Detect PPE");
           frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
           PPEBoundingBoxFrame panel = new PPEBoundingBoxFrame(result, image,
    confidence);
```

```
        panel.setPreferredSize(new Dimension(image.getWidth() / scale,
 image.getHeight() / scale));
        frame.setContentPane(panel);
        frame.pack();
        frame.setVisible(true);

    } catch (RekognitionException e) {
        e.printStackTrace();
        System.exit(1);
    } catch (Exception e) {
        e.printStackTrace();
    }
}

 public static byte[] getObjectBytes (S3Client s3, String bucketName, String
 keyName) {

     try {
         GetObjectRequest objectRequest = GetObjectRequest
             .builder()
             .key(keyName)
             .bucket(bucketName)
             .build();

         ResponseBytes<GetObjectResponse> objectBytes =
 s3.getObjectAsBytes(objectRequest);
         return objectBytes.asByteArray();

     } catch (S3Exception e) {
         System.err.println(e.awsErrorDetails().errorMessage());
         System.exit(1);
     }
     return null;
  }

 public PPEBoundingBoxFrame(DetectProtectiveEquipmentResponse ppeResult,
 BufferedImage bufImage, float requiredConfidence) {
     super();
     scale = 1; // increase to shrink image size.
     result = ppeResult;
     image = bufImage;
     confidence=requiredConfidence;
 }
```

```
// Draws the bounding box around the detected masks.
public void paintComponent(Graphics g) {
    float left = 0;
    float top = 0;
    int height = image.getHeight(this);
    int width = image.getWidth(this);
    int offset=20;

    Graphics2D g2d = (Graphics2D) g; // Create a Java2D version of g.

    // Draw the image.
    g2d.drawImage(image, 0, 0, width / scale, height / scale, this);
    g2d.setColor(new Color(0, 212, 0));

    // Iterate through detected persons and display bounding boxes.
    List<ProtectiveEquipmentPerson> persons = result.persons();
    for (ProtectiveEquipmentPerson person: persons) {

        List<ProtectiveEquipmentBodyPart> bodyParts=person.bodyParts();
        if (!bodyParts.isEmpty()){
            for (ProtectiveEquipmentBodyPart bodyPart: bodyParts) {
                List<EquipmentDetection>
equipmentDetections=bodyPart.equipmentDetections();
                for (EquipmentDetection item: equipmentDetections) {

                    String myType = item.type().toString();
                    if (myType.compareTo("FACE_COVER") ==0) {

                        // Draw green bounding box depending on mask coverage.
                        BoundingBox box =item.boundingBox();
                        left = width * box.left();
                        top = height * box.top();
                        Color maskColor=new Color( 0, 212, 0);

                        if (item.coversBodyPart().equals(false)) {
                            // red bounding box.
                            maskColor=new Color( 255, 0, 0);
                        }
                        g2d.setColor(maskColor);
                        g2d.drawRect(Math.round(left / scale), Math.round(top /
scale),
                                Math.round((width * box.width()) / scale),
Math.round((height * box.height())) / scale);
```

```
                        // Check confidence is > supplied confidence.
                        if (item.coversBodyPart().confidence() < confidence) {
                            // Draw a yellow bounding box inside face mask
 bounding box.
                            maskColor=new Color( 255, 255, 0);
                            g2d.setColor(maskColor);
                            g2d.drawRect(Math.round((left + offset) / scale),
                                    Math.round((top + offset) / scale),
                                    Math.round((width * box.width())- (offset *
 2 ))/ scale,
                                    Math.round((height * box.height()) -
( offset* 2)) / scale);
                        }
                    }
                }
            }
        }
    }
 }
 // snippet-end:[rekognition.java2.display_mask.main]
}
```

Python

En la función `main`, cambie lo siguiente:

- El valor de `photo` a la ruta y el nombre de archivo de un archivo de imagen local (PNG o JPEG).
- El valor de `confidence` al nivel de confianza deseado (50-100).
- Sustituya el valor de `profile_name` en la línea que crea la sesión de Rekognition por el nombre de su perfil de desarrollador.

```
#Copyright 2020 Amazon.com, Inc. or its affiliates. All Rights Reserved.
#PDX-License-Identifier: MIT-0 (For details, see https://github.com/awsdocs/
amazon-rekognition-developer-guide/blob/master/LICENSE-SAMPLECODE.)

import boto3
import io
from PIL import Image, ImageDraw, ExifTags, ImageColor

def detect_ppe(photo, confidence):
```

```
    fill_green='#00d400'
    fill_red='#ff0000'
    fill_yellow='#ffff00'
    line_width=3

    #open image and get image data from stream.
    image = Image.open(open(photo,'rb'))
    stream = io.BytesIO()
    image.save(stream, format=image.format)
    image_binary = stream.getvalue()
    imgWidth, imgHeight = image.size
    draw = ImageDraw.Draw(image)

    client=boto3.client('rekognition')

    response = client.detect_protective_equipment(Image={'Bytes': image_binary})

    for person in response['Persons']:

        found_mask=False

        for body_part in person['BodyParts']:
            ppe_items = body_part['EquipmentDetections']

            for ppe_item in ppe_items:
                #found a mask
                if ppe_item['Type'] == 'FACE_COVER':
                    fill_color=fill_green
                    found_mask=True
                    # check if mask covers face
                    if ppe_item['CoversBodyPart']['Value'] == False:
                        fill_color=fill='#ff0000'
                    # draw bounding box around mask
                    box = ppe_item['BoundingBox']
                    left = imgWidth * box['Left']
                    top = imgHeight * box['Top']
                    width = imgWidth * box['Width']
                    height = imgHeight * box['Height']
                    points = (
                            (left,top),
                            (left + width, top),
                            (left + width, top + height),
                            (left , top + height),
```

```
                            (left, top)
                        )
                    draw.line(points, fill=fill_color, width=line_width)

                     # Check if confidence is lower than supplied value
                    if ppe_item['CoversBodyPart']['Confidence'] < confidence:
                        #draw warning yellow bounding box within face mask
 bounding box
                        offset=line_width+ line_width
                        points = (
                                    (left+offset,top + offset),
                                    (left + width-offset, top+offset),
                                    ((left) + (width-offset), (top-offset) +
 (height)),
                                    (left+ offset , (top) + (height -offset)),
                                    (left + offset, top + offset)
                                )
                        draw.line(points, fill=fill_yellow, width=line_width)

        if found_mask==False:
            # no face mask found so draw red bounding box around body
            box = person['BoundingBox']
            left = imgWidth * box['Left']
            top = imgHeight * box['Top']
            width = imgWidth * box['Width']
            height = imgHeight * box['Height']
            points = (
                (left,top),
                (left + width, top),
                (left + width, top + height),
                (left , top + height),
                (left, top)
                )
            draw.line(points, fill=fill_red, width=line_width)

    image.show()

def main():
    photo='photo'
    confidence=80
    detect_ppe(photo, confidence)

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

CLI

En el siguiente ejemplo de la CLI, cambie el valor de los argumentos que se enumeran a continuación:

- El valor de `photo` a la ruta y el nombre de archivo de un archivo de imagen local (PNG o JPEG).
- El valor de `confidence` al nivel de confianza deseado (50-100).
- Sustituya el valor de `profile_name` en la línea que crea la sesión de Rekognition por el nombre de su perfil de desarrollador.

```
                                aws rekognition detect-protective-equipment
 --image "{"S3Object":{"Bucket":"amzn-s3-demo-bucket","Name":"image-name"}}" --
profile profile-name \
                                --summarization-attributes
 "{"MinConfidence":MinConfidenceNumber,"RequiredEquipmentTypes":["FACE_COVER"]}"
```

Si accede a la CLI en un dispositivo Windows, utilice comillas dobles en lugar de comillas simples y evite las comillas dobles internas con una barra invertida (es decir, \) para corregir cualquier error del analizador que pueda encontrar. Por ver un ejemplo, consulte lo siguiente:

```
                                aws rekognition detect-protective-equipment  --
image "{\"S3Object\":{\"Bucket\":\"amzn-s3-demo-bucket\",\"Name\":\"image-name
\"}}" \
                                --profile profile-name --summarization-
attributes "{\"MinConfidence\":MinConfidenceNumber,\"RequiredEquipmentTypes\":
[\"FACE_COVER\"]}"
```

Reconocimiento de famosos

Amazon Rekognition facilita a los clientes el reconocimiento automático de decenas de miles de personalidades conocidas en imágenes y vídeos mediante el machine learning. Los metadatos que proporciona la API de reconocimiento de famosos reducen considerablemente el esfuerzo manual repetitivo necesario para etiquetar el contenido y permiten buscarlo fácilmente.

La rápida proliferación de contenido de imágenes y vídeo hace que las empresas de medios de comunicación a menudo tengan dificultades para organizar, buscar y utilizar sus catálogos de medios a gran escala. Los canales de noticias y las emisoras deportivas a menudo necesitan encontrar imágenes y vídeos rápidamente para poder responder a los acontecimientos actuales y crear una programación relevante. La falta de metadatos dificulta estas tareas, pero con Amazon Rekognition puede etiquetar automáticamente grandes volúmenes de contenido nuevo o archivado para que un conjunto completo de personas famosas a nivel internacional, como actores, deportistas y creadores de contenido en línea, puedan buscarlo fácilmente.

El reconocimiento de personas famosas de Amazon Rekognition está diseñado para usarse exclusivamente en los casos en los que espere que aparezca una celebridad conocida en una imagen o un vídeo. Para obtener más información acerca de cómo reconocer rostros que no son famosos, consulte Búsqueda de rostros en una colección.

> **Note**
>
> Si eres una celebridad y no quieres que te incluyan en esta función, ponte en contacto con AWS Support o envía un correo electrónico `<rekognition-celebrity-opt-outa @amazon .com>`.

Temas

- Reconocimiento de famosos comparado con la búsqueda de rostros
- Reconocimiento de famosos en una imagen
- Reconocimiento de famosos en un vídeo almacenado
- Obtención de información sobre un famoso

Reconocimiento de famosos comparado con la búsqueda de rostros

Amazon Rekognition ofrece funcionalidad de reconocimiento de famosos y de reconocimiento facial. Existen algunas diferencias esenciales entre estas funcionalidades en términos de sus casos de uso y prácticas recomendadas.

El reconocimiento de famosos se suministra previamente entrenado para reconocer a más de cientos de miles de famosos en ámbitos como los deportes, los medios de comunicación, la política o los negocios. Esta funcionalidad está diseñada para ayudarle a buscar en grandes volúmenes de imágenes o vídeos con el fin de identificar un pequeño conjunto que es probable que contenga a un determinado famoso. No está diseñado para hallar coincidencias entre rostros de distintas personas que no son famosas. En aquellas situaciones en que sea importante la precisión de la coincidencia de famosos, recomendamos utilizar también a operadores humanos para revisar este conjunto de contenido marcado, con el fin de garantizar un alto nivel de precisión, así como la aplicación debida del criterio humano. El reconocimiento de famosos no debe utilizarse de ningún modo que pueda afectar negativamente a las libertades civiles.

Por otra parte, el reconocimiento facial es una funcionalidad más general que le permite crear sus propias colecciones de rostros con sus propios vectores faciales para comprobar identidades o buscar a cualquier persona, no solo a famosos. El reconocimiento facial se puede utilizar para aplicaciones como la autenticación para el acceso a edificios, la seguridad pública o las redes sociales. En todos estos casos, recomendamos aplicar las prácticas recomendadas, los umbrales de confianza adecuados (incluido el 99 % en los casos de uso con fines de seguridad pública) y la revisión humana en aquellas situaciones en que sea importante la precisión de la coincidencia.

Para obtener más información, consulte Búsqueda de rostros en una colección.

Reconocimiento de famosos en una imagen

Para reconocer a famosos en imágenes y obtener más información sobre los famosos reconocidos, utilice la operación API sin almacenamiento RecognizeCelebrities. Por ejemplo, en las redes sociales o en los sectores de noticias y entretenimiento, donde el factor tiempo de recopilación de la información puede ser crítico, puede utilizar la operación `RecognizeCelebrities` para identificar hasta 64 famosos en una imagen y devolver enlaces a páginas web de famosos, si están disponibles. Amazon Rekognition no recuerda la imagen en la que detectó a un famoso. La aplicación debe almacenar esta información.

Si no ha almacenado la información adicional para un famoso devuelta por `RecognizeCelebrities` y desea evitar tener que volver a analizar una imagen para obtenerla, utilice GetCelebrityInfo. Para llamar a `GetCelebrityInfo`, necesita el identificador único que Amazon Rekognition asigna a cada famoso. El identificador se devuelve como parte de la respuesta `RecognizeCelebrities` para cada famoso reconocido en una imagen.

Si tiene una amplia gama de imágenes para procesar a fin de reconocer famosos, considere la posibilidad de utilizar AWS Batch para procesar las llamadas a `RecognizeCelebrities` en lotes en segundo plano. Cuando añadas una imagen nueva a tu colección, podrás usar una AWS Lambda función para reconocer a los famosos llamando `RecognizeCelebrities` mientras la imagen se carga en un compartimento de S3.

¿Llamando RecognizeCelebrities

Puede proporcionar la imagen de entrada como una matriz de bytes de imagen (bytes de imagen codificados en base64) o como un objeto de Amazon S3, mediante AWS Command Line Interface (AWS CLI) o el SDK de AWS. En el AWS CLI procedimiento, debe cargar una imagen en formato.jpg o .png a un bucket de S3. En los procedimientos del AWS SDK, se utiliza una imagen que se carga desde el sistema de archivos local. Para obtener información sobre recomendaciones de imagen de entrada, consulte Trabajar con imágenes.

Para ejecutar este procedimiento, necesitará un archivo de imagen que contenga uno o varios rostros de famosos.

Reconocimiento de famosos en una imagen

1. Si aún no lo ha hecho:

   a. Cree o actualice un usuario con los permisos `AmazonRekognitionFullAccess` y `AmazonS3ReadOnlyAccess`. Para obtener más información, consulte Paso 1: Configurar una cuenta de AWS y crear un usuario.

   b. Instala y configura el AWS CLI y el AWS SDKs. Para obtener más información, consulte Paso 2: Configure y AWS CLI AWS SDKs.

2. Consulte los siguientes ejemplos para llamar a la operación `RecognizeCelebrities`.

   Java

   En este ejemplo se muestra información acerca de los famosos que se detectan en una imagen.

Cambie el valor de `photo` por la ruta y el nombre de un archivo de imagen que contenga uno o más rostros de famosos.

```
//Copyright 2018 Amazon.com, Inc. or its affiliates. All Rights Reserved.
//PDX-License-Identifier: MIT-0 (For details, see https://github.com/awsdocs/
amazon-rekognition-developer-guide/blob/master/LICENSE-SAMPLECODE.)

package aws.example.rekognition.image;
import com.amazonaws.services.rekognition.AmazonRekognition;
import com.amazonaws.services.rekognition.AmazonRekognitionClientBuilder;
import com.amazonaws.services.rekognition.model.Image;
import com.amazonaws.services.rekognition.model.BoundingBox;
import com.amazonaws.services.rekognition.model.Celebrity;
import com.amazonaws.services.rekognition.model.RecognizeCelebritiesRequest;
import com.amazonaws.services.rekognition.model.RecognizeCelebritiesResult;
import java.io.File;
import java.io.FileInputStream;
import java.io.InputStream;
import java.nio.ByteBuffer;
import com.amazonaws.util.IOUtils;
import java.util.List;


public class RecognizeCelebrities {

   public static void main(String[] args) {
      String photo = "moviestars.jpg";

      AmazonRekognition rekognitionClient =
 AmazonRekognitionClientBuilder.defaultClient();

      ByteBuffer imageBytes=null;
      try (InputStream inputStream = new FileInputStream(new File(photo))) {
         imageBytes = ByteBuffer.wrap(IOUtils.toByteArray(inputStream));
      }
      catch(Exception e)
      {
          System.out.println("Failed to load file " + photo);
          System.exit(1);
      }
```

```
      RecognizeCelebritiesRequest request = new RecognizeCelebritiesRequest()
         .withImage(new Image()
         .withBytes(imageBytes));

      System.out.println("Looking for celebrities in image " + photo + "\n");

      RecognizeCelebritiesResult
 result=rekognitionClient.recognizeCelebrities(request);

      //Display recognized celebrity information
      List<Celebrity> celebs=result.getCelebrityFaces();
      System.out.println(celebs.size() + " celebrity(s) were recognized.\n");

      for (Celebrity celebrity: celebs) {
          System.out.println("Celebrity recognized: " + celebrity.getName());
          System.out.println("Celebrity ID: " + celebrity.getId());
          BoundingBox boundingBox=celebrity.getFace().getBoundingBox();
          System.out.println("position: " +
             boundingBox.getLeft().toString() + " " +
             boundingBox.getTop().toString());
          System.out.println("Further information (if available):");
          for (String url: celebrity.getUrls()){
             System.out.println(url);
          }
          System.out.println();
       }
       System.out.println(result.getUnrecognizedFaces().size() + " face(s) were
 unrecognized.");
    }
}
```

Java V2

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

```
//snippet-start:[rekognition.java2.recognize_celebs.import]
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.core.SdkBytes;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
```

```
import java.io.InputStream;
import java.util.List;
import
 software.amazon.awssdk.services.rekognition.model.RecognizeCelebritiesRequest;
import
 software.amazon.awssdk.services.rekognition.model.RecognizeCelebritiesResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import software.amazon.awssdk.services.rekognition.model.Image;
import software.amazon.awssdk.services.rekognition.model.Celebrity;
//snippet-end:[rekognition.java2.recognize_celebs.import]

/**
* Before running this Java V2 code example, set up your development environment,
 including your credentials.
*
* For more information, see the following documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
*/
public class RecognizeCelebrities {

 public static void main(String[] args) {

     final String usage = "\n" +
         "Usage: " +
         "   <sourceImage>\n\n" +
         "Where:\n" +
         "   sourceImage - The path to the image (for example, C:\\AWS\
\pic1.png). \n\n";

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

     String sourceImage = args[0];
     Region region = Region.US_EAST_1;
     RekognitionClient rekClient = RekognitionClient.builder()
         .region(region)
         .credentialsProvider(ProfileCredentialsProvider.create("profile-name"))
         .build();

     System.out.println("Locating celebrities in " + sourceImage);
```

```
    recognizeAllCelebrities(rekClient, sourceImage);
    rekClient.close();
}

// snippet-start:[rekognition.java2.recognize_celebs.main]
public static void recognizeAllCelebrities(RekognitionClient rekClient, String
sourceImage) {

    try {
        InputStream sourceStream = new FileInputStream(sourceImage);
        SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);
        Image souImage = Image.builder()
            .bytes(sourceBytes)
            .build();

        RecognizeCelebritiesRequest request =
RecognizeCelebritiesRequest.builder()
            .image(souImage)
            .build();

        RecognizeCelebritiesResponse result =
rekClient.recognizeCelebrities(request) ;
        List<Celebrity> celebs=result.celebrityFaces();
        System.out.println(celebs.size() + " celebrity(s) were recognized.\n");
        for (Celebrity celebrity: celebs) {
            System.out.println("Celebrity recognized: " + celebrity.name());
            System.out.println("Celebrity ID: " + celebrity.id());

            System.out.println("Further information (if available):");
            for (String url: celebrity.urls()){
                System.out.println(url);
            }
            System.out.println();
        }
        System.out.println(result.unrecognizedFaces().size() + " face(s) were
unrecognized.");

    } catch (RekognitionException | FileNotFoundException e) {
        System.out.println(e.getMessage());
        System.exit(1);
    }
}
// snippet-end:[rekognition.java2.recognize_celebs.main]
```

```
}
```

AWS CLI

Este AWS CLI comando muestra el resultado JSON de la operación `recognize-celebrities` CLI.

Cambie `amzn-s3-demo-bucket` por el nombre del bucket de Amazon S3 que contiene una imagen. Cambie el valor de `input.jpg` por el nombre de un archivo de imagen que contenga uno o más rostros de famosos.

Sustituya el valor de `profile_name` de por el nombre de su perfil de desarrollador.

```
aws rekognition recognize-celebrities \
  --image "S3Object={Bucket=amzn-s3-demo-bucket,Name=input.jpg}"
```

Si accede a la CLI en un dispositivo Windows, utilice comillas dobles en lugar de comillas simples y evite las comillas dobles internas con una barra invertida (es decir, \) para corregir cualquier error del analizador que pueda encontrar. Por ver un ejemplo, consulte lo siguiente:

```
                                    aws rekognition recognize-celebrities --
image \
                                    "{\"S3Object\":{\"Bucket\":\"amzn-s3-demo-
bucket\",\"Name\":\"image-name\"}}" --profile profile-name
```

Python

En este ejemplo se muestra información acerca de los famosos que se detectan en una imagen.

Cambie el valor de `photo` por la ruta y el nombre de un archivo de imagen que contenga uno o más rostros de famosos.

Sustituya el valor de `profile_name` en la línea que crea la sesión de Rekognition por el nombre de su perfil de desarrollador.

```
#Copyright 2018 Amazon.com, Inc. or its affiliates. All Rights Reserved.
```

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

import boto3

def recognize_celebrities(photo):

    session = boto3.Session(profile_name='profile-name')
    client = session.client('rekognition')

    with open(photo, 'rb') as image:
        response = client.recognize_celebrities(Image={'Bytes': image.read()})

    print('Detected faces for ' + photo)
    for celebrity in response['CelebrityFaces']:
        print('Name: ' + celebrity['Name'])
        print('Id: ' + celebrity['Id'])
        print('KnownGender: ' + celebrity['KnownGender']['Type'])
        print('Smile: ' + str(celebrity['Face']['Smile']['Value']))
        print('Position:')
        print('   Left: ' + '{:.2f}'.format(celebrity['Face']['BoundingBox']
['Height']))
        print('   Top: ' + '{:.2f}'.format(celebrity['Face']['BoundingBox']
['Top']))
        print('Info')
        for url in celebrity['Urls']:
            print('   ' + url)
        print()
    return len(response['CelebrityFaces'])

def main():
    photo = 'photo-name'
    celeb_count = recognize_celebrities(photo)
    print("Celebrities detected: " + str(celeb_count))

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

Node.Js

En este ejemplo se muestra información acerca de los famosos que se detectan en una imagen.

Cambie el valor de `photo` por la ruta y el nombre de un archivo de imagen que contenga uno o más rostros de famosos. Cambie el valor de `amzn-s3-demo-bucket` por el nombre del bucket de S3 que contiene el archivo de imagen proporcionado. Sustituya el valor de `REGION` por el nombre de la región asociada a su usuario. Sustituya el valor de `profile_name` en la línea que crea la sesión de Rekognition por el nombre de su perfil de desarrollador.

```
// Import required AWS SDK clients and commands for Node.js
import { RecognizeCelebritiesCommand } from  "@aws-sdk/client-rekognition";
import  { RekognitionClient } from "@aws-sdk/client-rekognition";

// Set the AWS Region.
const REGION = "region-name"; //e.g. "us-east-1"
const profileName = "profile-name";

// Create SNS service object.
const rekogClient = new RekognitionClient({region: REGION,
  credentials: fromIni({profile: profileName,}),
});

const bucket = 'bucket-name'
const photo = 'photo-name'

// Set params
const params = {
    Image: {
      S3Object: {
        Bucket: bucket,
        Name: photo
      },
    },
  }

const recognize_celebrity = async() => {
    try {
        const response = await rekogClient.send(new
 RecognizeCelebritiesCommand(params));
        console.log(response.Labels)
        response.CelebrityFaces.forEach(celebrity =>{
            console.log(`Name: ${celebrity.Name}`)
            console.log(`ID: ${celebrity.Id}`)
            console.log(`KnownGender: ${celebrity.KnownGender.Type}`)
            console.log(`Smile: ${celebrity.Smile}`)
```

```
            console.log('Position: ')
            console.log(`   Left: ${celebrity.Face.BoundingBox.Height}`)
            console.log(`  Top : ${celebrity.Face.BoundingBox.Top}`)

        })
        return response.length; // For unit tests.
      } catch (err) {
        console.log("Error", err);
      }
}

recognize_celebrity()
```

.NET

En este ejemplo se muestra información acerca de los famosos que se detectan en una imagen.

Cambie el valor de `photo` por la ruta y el nombre de un archivo de imagen que contenga uno o más rostros de famosos (en formato .jpg o .png).

```
//Copyright 2018 Amazon.com, Inc. or its affiliates. All Rights Reserved.
//PDX-License-Identifier: MIT-0 (For details, see https://github.com/awsdocs/
amazon-rekognition-developer-guide/blob/master/LICENSE-SAMPLECODE.)

using System;
using System.IO;
using Amazon.Rekognition;
using Amazon.Rekognition.Model;

public class CelebritiesInImage
{
    public static void Example()
    {
        String photo = "moviestars.jpg";

        AmazonRekognitionClient rekognitionClient = new
 AmazonRekognitionClient();

        RecognizeCelebritiesRequest recognizeCelebritiesRequest = new
 RecognizeCelebritiesRequest();
```

```
       Amazon.Rekognition.Model.Image img = new
Amazon.Rekognition.Model.Image();
       byte[] data = null;
       try
       {
           using (FileStream fs = new FileStream(photo, FileMode.Open,
FileAccess.Read))
           {
               data = new byte[fs.Length];
               fs.Read(data, 0, (int)fs.Length);
           }
       }
       catch(Exception)
       {
           Console.WriteLine("Failed to load file " + photo);
           return;
       }

       img.Bytes = new MemoryStream(data);
       recognizeCelebritiesRequest.Image = img;

       Console.WriteLine("Looking for celebrities in image " + photo + "\n");

       RecognizeCelebritiesResponse recognizeCelebritiesResponse =
rekognitionClient.RecognizeCelebrities(recognizeCelebritiesRequest);

       Console.WriteLine(recognizeCelebritiesResponse.CelebrityFaces.Count + "
celebrity(s) were recognized.\n");
       foreach (Celebrity celebrity in
recognizeCelebritiesResponse.CelebrityFaces)
       {
           Console.WriteLine("Celebrity recognized: " + celebrity.Name);
           Console.WriteLine("Celebrity ID: " + celebrity.Id);
           BoundingBox boundingBox = celebrity.Face.BoundingBox;
           Console.WriteLine("position: " +
              boundingBox.Left + " " + boundingBox.Top);
           Console.WriteLine("Further information (if available):");
           foreach (String url in celebrity.Urls)
               Console.WriteLine(url);
       }
       Console.WriteLine(recognizeCelebritiesResponse.UnrecognizedFaces.Count +
" face(s) were unrecognized.");
   }
```

```
}
```

3. Registre el valor de una de las celebridades IDs que se muestran. Lo necesitará en Obtención de información sobre un famoso.

RecognizeCelebrities solicitud de operación

La entrada de `RecognizeCelebrities` es una imagen. En este ejemplo, la imagen se transfiere como bytes de imagen. Para obtener más información, consulte Trabajar con imágenes.

```
{
    "Image": {
        "Bytes": "/AoSiyvFpm....."
    }
}
```

RecognizeCelebrities respuesta de operación

A continuación se ofrece un ejemplo de JSON de la salida y la entrada de `RecognizeCelebrities`.

`RecognizeCelebrities` devuelve una matriz de famosos reconocidos y una matriz de rostros no reconocidos, tal y como se muestra en el ejemplo siguiente. En el ejemplo de , observe lo siguiente:

- Famosos reconocidos: `Celebrities` es una matriz de famosos reconocidos. Cada objeto de celebridad de la matriz contiene el nombre de la celebridad y una lista que URLs apunta a contenido relacionado, por ejemplo, el enlace de IMDB o Wikidata de la celebridad. Amazon Rekognition ComparedFacedevuelve un objeto que la aplicación puede utilizar para determinar dónde aparece el rostro de la celebridad en la imagen y un identificador único de la celebridad. Utilice el identificador único para recuperar información sobre el famoso más adelante con la operación API GetCelebrityInfo.
- Rostros no reconocidos: `UnrecognizedFaces` es una matriz de rostros que no coinciden con ningún famoso conocido. Cada objeto ComparedFace de la matriz contiene un cuadro delimitador (además de otra información) que puede utilizar para localizar el rostro en la imagen.

```
{
    "CelebrityFaces": [{
```

```
        "Face": {
            "BoundingBox": {
                "Height": 0.617123007774353,
                "Left": 0.15641026198863983,
                "Top": 0.10864841192960739,
                "Width": 0.3641025722026825
            },
            "Confidence": 99.99589538574219,
            "Emotions": [{
                "Confidence": 96.3981749057023,
                "Type": "Happy"
                }
            ],
            "Landmarks": [{
                "Type": "eyeLeft",
                "X": 0.2837241291999817,
                "Y": 0.3637104034423828
            }, {
                "Type": "eyeRight",
                "X": 0.4091649055480957,
                "Y": 0.37378931045532227
            }, {
                "Type": "nose",
                "X": 0.35267341136932373,
                "Y": 0.49657556414604187
            }, {
                "Type": "mouthLeft",
                "X": 0.2786353826522827,
                "Y": 0.5455248355865479
            }, {
                "Type": "mouthRight",
                "X": 0.39566439390182495,
                "Y": 0.5597742199897766
            }],
            "Pose": {
                "Pitch": -7.749263763427734,
                "Roll": 2.004552125930786,
                "Yaw": 9.012002944946289
            },
            "Quality": {
                "Brightness": 32.69192123413086,
                "Sharpness": 99.9305191040039
            },
            "Smile": {
```

```
            "Confidence": 95.45394855702342,
            "Value": True
            }
        },
        "Id": "3Ir0du6",
        "KnownGender": {
            "Type": "Male"
        },
        "MatchConfidence": 98.0,
        "Name": "Jeff Bezos",
        "Urls": ["www.imdb.com/name/nm1757263"]
    }],
    "OrientationCorrection": "NULL",
    "UnrecognizedFaces": [{
        "BoundingBox": {
            "Height": 0.5345501899719238,
            "Left": 0.48461538553237915,
            "Top": 0.16949152946472168,
            "Width": 0.3153846263885498
        },
        "Confidence": 99.92860412597656,
        "Landmarks": [{
            "Type": "eyeLeft",
            "X": 0.5863404870033264,
            "Y": 0.36940744519233704
        }, {
            "Type": "eyeRight",
            "X": 0.6999204754829407,
            "Y": 0.3769848346710205
        }, {
            "Type": "nose",
            "X": 0.6349524259567261,
            "Y": 0.4804527163505554
        }, {
            "Type": "mouthLeft",
            "X": 0.5872702598571777,
            "Y": 0.5535582304000854
        }, {
            "Type": "mouthRight",
            "X": 0.6952020525932312,
            "Y": 0.5600858926773071
        }],
        "Pose": {
            "Pitch": -7.386096477508545,
```

```
            "Roll": 2.304218292236328,
            "Yaw": -6.175624370574951
        },
        "Quality": {
            "Brightness": 37.16635513305664,
            "Sharpness": 99.9305191040039
        },
        "Smile": {
            "Confidence": 95.45394855702342,
            "Value": True
        }
    }]
}
```

Reconocimiento de famosos en un vídeo almacenado

El reconocimiento de famosos de Amazon Rekognition Video en los vídeos almacenados es una operación asíncrona. Para reconocer a las celebridades en un vídeo almacenado, utilice esta opción StartCelebrityRecognitionpara iniciar el análisis del vídeo. Amazon Rekognition Video publica el estado de finalización de una operación de análisis de vídeo en un tema de Amazon Simple Notification Service. Si el análisis de vídeo es correcto, llame a GetCelebrityRecognition para obtener los resultados del análisis. Para obtener más información sobre cómo iniciar el análisis de vídeo y obtener los resultados, consulte Cómo llamar a las operaciones de Amazon Rekognition Video.

Este procedimiento amplía el código de Análisis de un vídeo almacenado en un bucket de Amazon S3 con Java o Python (SDK), que utiliza una cola de Amazon SQS para obtener el estado de realización de una solicitud de análisis de vídeo. Para ejecutar este procedimiento, necesitará un archivo de vídeo que contenga uno o varios rostros de famosos.

Para detectar famosos en un vídeo almacenado en un bucket de Amazon S3 (SDK)

1. Realice Análisis de un vídeo almacenado en un bucket de Amazon S3 con Java o Python (SDK).
2. Añada el código siguiente a la clase `VideoDetect` que ha creado en el paso 1.

   Java

```
        //Copyright 2018 Amazon.com, Inc. or its affiliates. All Rights
 Reserved.
        //PDX-License-Identifier: MIT-0 (For details, see https://github.com/
awsdocs/amazon-rekognition-developer-guide/blob/master/LICENSE-SAMPLECODE.)
```

```
    //
Celebrities========================================================================
    private static void StartCelebrityDetection(String bucket, String video)
throws Exception{

          NotificationChannel channel= new NotificationChannel()
                  .withSNSTopicArn(snsTopicArn)
                  .withRoleArn(roleArn);

         StartCelebrityRecognitionRequest req = new
StartCelebrityRecognitionRequest()
               .withVideo(new Video()
                     .withS3Object(new S3Object()
                           .withBucket(bucket)
                           .withName(video)))
               .withNotificationChannel(channel);



         StartCelebrityRecognitionResult startCelebrityRecognitionResult =
rek.startCelebrityRecognition(req);
         startJobId=startCelebrityRecognitionResult.getJobId();

      }

      private static void GetCelebrityDetectionResults() throws Exception{

         int maxResults=10;
         String paginationToken=null;
         GetCelebrityRecognitionResult celebrityRecognitionResult=null;

         do{
            if (celebrityRecognitionResult !=null){
               paginationToken = celebrityRecognitionResult.getNextToken();
            }
            celebrityRecognitionResult = rek.getCelebrityRecognition(new
GetCelebrityRecognitionRequest()
                  .withJobId(startJobId)
                  .withNextToken(paginationToken)
                  .withSortBy(CelebrityRecognitionSortBy.TIMESTAMP)
                  .withMaxResults(maxResults));
```

```
            System.out.println("File info for page");
            VideoMetadata
videoMetaData=celebrityRecognitionResult.getVideoMetadata();

            System.out.println("Format: " + videoMetaData.getFormat());
            System.out.println("Codec: " + videoMetaData.getCodec());
            System.out.println("Duration: " +
videoMetaData.getDurationMillis());
            System.out.println("FrameRate: " + videoMetaData.getFrameRate());

            System.out.println("Job");

            System.out.println("Job status: " +
celebrityRecognitionResult.getJobStatus());


            //Show celebrities
            List<CelebrityRecognition> celebs=
celebrityRecognitionResult.getCelebrities();

            for (CelebrityRecognition celeb: celebs) {
               long seconds=celeb.getTimestamp()/1000;
               System.out.print("Sec: " + Long.toString(seconds) + " ");
               CelebrityDetail details=celeb.getCelebrity();
               System.out.println("Name: " + details.getName());
               System.out.println("Id: " + details.getId());
               System.out.println();
            }
          } while (celebrityRecognitionResult !=null &&
celebrityRecognitionResult.getNextToken() != null);

       }
```

En la función `main`, reemplace la línea:

```
       StartLabelDetection(amzn-s3-demo-bucket, video);

       if (GetSQSMessageSuccess()==true)
        GetLabelDetectionResults();
```

por:

```
        StartCelebrityDetection(amzn-s3-demo-bucket, video);

        if (GetSQSMessageSuccess()==true)
         GetCelebrityDetectionResults();
```

Java V2

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

```
//snippet-start:[rekognition.java2.recognize_video_celebrity.import]
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.S3Object;
import software.amazon.awssdk.services.rekognition.model.NotificationChannel;
import software.amazon.awssdk.services.rekognition.model.Video;
import
 software.amazon.awssdk.services.rekognition.model.StartCelebrityRecognitionResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import
 software.amazon.awssdk.services.rekognition.model.CelebrityRecognitionSortBy;
import software.amazon.awssdk.services.rekognition.model.VideoMetadata;
import software.amazon.awssdk.services.rekognition.model.CelebrityRecognition;
import software.amazon.awssdk.services.rekognition.model.CelebrityDetail;
import
 software.amazon.awssdk.services.rekognition.model.StartCelebrityRecognitionRequest;
import
 software.amazon.awssdk.services.rekognition.model.GetCelebrityRecognitionRequest;
import
 software.amazon.awssdk.services.rekognition.model.GetCelebrityRecognitionResponse;
import java.util.List;
//snippet-end:[rekognition.java2.recognize_video_celebrity.import]

/**
*  To run this code example, ensure that you perform the Prerequisites as stated
 in the Amazon Rekognition Guide:
*  https://docs.aws.amazon.com/rekognition/latest/dg/video-analyzing-with-
sqs.html
*
```

```
* Also, ensure that set up your development environment, including your
 credentials.
*
* For information, see this documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
*/

public class RecognizeCelebritiesVideo {

private static String startJobId ="";

public static void main(String[] args) {

   final String usage = "\n" +
       "Usage: " +
       "   <bucket> <video> <topicArn> <roleArn>\n\n" +
       "Where:\n" +
       "   bucket - The name of the bucket in which the video is located (for
 example, (for example, amzn-s3-demo-bucket). \n\n"+
       "   video - The name of video (for example, people.mp4). \n\n" +
       "   topicArn - The ARN of the Amazon Simple Notification Service (Amazon
 SNS) topic. \n\n" +
       "   roleArn - The ARN of the AWS Identity and Access Management (IAM)
 role to use. \n\n" ;

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

   String bucket = args[0];
   String video = args[1];
   String topicArn = args[2];
   String roleArn = args[3];
   Region region = Region.US_EAST_1;
   RekognitionClient rekClient = RekognitionClient.builder()
       .region(region)
       .credentialsProvider(ProfileCredentialsProvider.create("profile-name"))
       .build();

  NotificationChannel channel = NotificationChannel.builder()
       .snsTopicArn(topicArn)
```

```
        .roleArn(roleArn)
        .build();

  StartCelebrityDetection(rekClient, channel, bucket, video);
  GetCelebrityDetectionResults(rekClient);
  System.out.println("This example is done!");
  rekClient.close();
}

// snippet-start:[rekognition.java2.recognize_video_celebrity.main]
public static void StartCelebrityDetection(RekognitionClient rekClient,
                                            NotificationChannel channel,
                                            String bucket,
                                            String video){
   try {
       S3Object s3Obj = S3Object.builder()
           .bucket(bucket)
           .name(video)
           .build();

       Video vidOb = Video.builder()
           .s3Object(s3Obj)
           .build();

       StartCelebrityRecognitionRequest recognitionRequest =
 StartCelebrityRecognitionRequest.builder()
           .jobTag("Celebrities")
           .notificationChannel(channel)
           .video(vidOb)
           .build();

       StartCelebrityRecognitionResponse startCelebrityRecognitionResult =
 rekClient.startCelebrityRecognition(recognitionRequest);
       startJobId = startCelebrityRecognitionResult.jobId();

   } catch(RekognitionException e) {
       System.out.println(e.getMessage());
       System.exit(1);
   }
}

public static void GetCelebrityDetectionResults(RekognitionClient rekClient) {

   try {
```

```
        String paginationToken=null;
        GetCelebrityRecognitionResponse recognitionResponse = null;
        boolean finished = false;
        String status;
        int yy=0 ;

        do{
            if (recognitionResponse !=null)
                paginationToken = recognitionResponse.nextToken();

            GetCelebrityRecognitionRequest recognitionRequest =
GetCelebrityRecognitionRequest.builder()
                .jobId(startJobId)
                .nextToken(paginationToken)
                .sortBy(CelebrityRecognitionSortBy.TIMESTAMP)
                .maxResults(10)
                .build();

            // Wait until the job succeeds
            while (!finished) {
                recognitionResponse =
rekClient.getCelebrityRecognition(recognitionRequest);
                status = recognitionResponse.jobStatusAsString();

                if (status.compareTo("SUCCEEDED") == 0)
                    finished = true;
                else {
                    System.out.println(yy + " status is: " + status);
                    Thread.sleep(1000);
                }
                yy++;
            }

            finished = false;

            // Proceed when the job is done - otherwise VideoMetadata is null.
            VideoMetadata videoMetaData=recognitionResponse.videoMetadata();
            System.out.println("Format: " + videoMetaData.format());
            System.out.println("Codec: " + videoMetaData.codec());
            System.out.println("Duration: " + videoMetaData.durationMillis());
            System.out.println("FrameRate: " + videoMetaData.frameRate());
            System.out.println("Job");

            List<CelebrityRecognition> celebs= recognitionResponse.celebrities();
```

```
            for (CelebrityRecognition celeb: celebs) {
                long seconds=celeb.timestamp()/1000;
                System.out.print("Sec: " + seconds + " ");
                CelebrityDetail details=celeb.celebrity();
                System.out.println("Name: " + details.name());
                System.out.println("Id: " + details.id());
                System.out.println();
            }

        } while (recognitionResponse.nextToken() != null);

    } catch(RekognitionException | InterruptedException e) {
        System.out.println(e.getMessage());
        System.exit(1);
    }
}
// snippet-end:[rekognition.java2.recognize_video_celebrity.main]
}
```

Python

```
#Copyright 2018 Amazon.com, Inc. or its affiliates. All Rights Reserved.
#PDX-License-Identifier: MIT-0 (For details, see https://github.com/awsdocs/
amazon-rekognition-developer-guide/blob/master/LICENSE-SAMPLECODE.)

    # ============== Celebrities ===============
    def StartCelebrityDetection(self):
        response=self.rek.start_celebrity_recognition(Video={'S3Object':
 {'Bucket': self.bucket, 'Name': self.video}},
            NotificationChannel={'RoleArn': self.roleArn, 'SNSTopicArn':
 self.snsTopicArn})

        self.startJobId=response['JobId']
        print('Start Job Id: ' + self.startJobId)

    def GetCelebrityDetectionResults(self):
        maxResults = 10
        paginationToken = ''
        finished = False

        while finished == False:
            response = self.rek.get_celebrity_recognition(JobId=self.startJobId,
                                                MaxResults=maxResults,
```

```
                                                NextToken=paginationToken)

            print(response['VideoMetadata']['Codec'])
            print(str(response['VideoMetadata']['DurationMillis']))
            print(response['VideoMetadata']['Format'])
            print(response['VideoMetadata']['FrameRate'])

            for celebrityRecognition in response['Celebrities']:
                print('Celebrity: ' +
                    str(celebrityRecognition['Celebrity']['Name']))
                print('Timestamp: ' + str(celebrityRecognition['Timestamp']))
                print()

            if 'NextToken' in response:
                paginationToken = response['NextToken']
            else:
                finished = True
```

En la función `main`, reemplace las líneas:

```
    analyzer.StartLabelDetection()
    if analyzer.GetSQSMessageSuccess()==True:
        analyzer.GetLabelDetectionResults()
```

por:

```
    analyzer.StartCelebrityDetection()
    if analyzer.GetSQSMessageSuccess()==True:
        analyzer.GetCelebrityDetectionResults()
```

Node.JS

En el siguiente ejemplo de código de Node.Js, sustituya el valor de `amzn-s3-demo-bucket` por el nombre del bucket de S3 que contiene el vídeo y el valor de `videoName` por el nombre del archivo de vídeo. También tendrá que reemplazar el valor de `roleArn` por el Arn asociado a su rol de servicio de IAM. Por último, sustituya el valor de `region` por el nombre de la región de operación asociada a su cuenta. Sustituya el valor de `profile_name` en la línea que crea la sesión de Rekognition por el nombre de su perfil de desarrollador.

```
//Copyright 2018 Amazon.com, Inc. or its affiliates. All Rights Reserved.
```

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

// Import required AWS SDK clients and commands for Node.js
import { CreateQueueCommand, GetQueueAttributesCommand, GetQueueUrlCommand,
  SetQueueAttributesCommand, DeleteQueueCommand, ReceiveMessageCommand,
 DeleteMessageCommand } from  "@aws-sdk/client-sqs";
import {CreateTopicCommand, SubscribeCommand, DeleteTopicCommand } from "@aws-
sdk/client-sns";
import  { SQSClient } from "@aws-sdk/client-sqs";
import  { SNSClient } from "@aws-sdk/client-sns";
import  { RekognitionClient, StartLabelDetectionCommand,
 GetLabelDetectionCommand,
  StartCelebrityRecognitionCommand, GetCelebrityRecognitionCommand} from "@aws-
sdk/client-rekognition";
import { stdout } from "process";
import {fromIni} from '@aws-sdk/credential-providers';

// Set the AWS Region.
const REGION = "region-name"; //e.g. "us-east-1"
// Set the profile name
const profileName = "profile-name"
// Name the collection
// Create SNS service object.
const sqsClient = new SQSClient({ region: REGION,
  credentials: fromIni({profile: profileName,}), });
const snsClient = new SNSClient({ region: REGION,
  credentials: fromIni({profile: profileName,}), });
const rekClient = new RekognitionClient({region: REGION,
  credentials: fromIni({profile: profileName,}),
});

// Set bucket and video variables
const bucket = "bucket-name";
const videoName = "video-name";
const roleArn = "role-arn"
var startJobId = ""

var ts = Date.now();
const snsTopicName = "AmazonRekognitionExample" + ts;
const snsTopicParams = {Name: snsTopicName}
const sqsQueueName = "AmazonRekognitionQueue-" + ts;

 // Set the parameters
```

```
 const sqsParams = {
  QueueName: sqsQueueName, //SQS_QUEUE_URL
  Attributes: {
    DelaySeconds: "60", // Number of seconds delay.
    MessageRetentionPeriod: "86400", // Number of seconds delay.
  },
};

const createTopicandQueue = async () => {
  try {
    // Create SNS topic
    const topicResponse = await snsClient.send(new
 CreateTopicCommand(snsTopicParams));
    const topicArn = topicResponse.TopicArn
    console.log("Success", topicResponse);
    // Create SQS Queue
    const sqsResponse = await sqsClient.send(new
 CreateQueueCommand(sqsParams));
    console.log("Success", sqsResponse);
    const sqsQueueCommand = await sqsClient.send(new
 GetQueueUrlCommand({QueueName: sqsQueueName}))
    const sqsQueueUrl = sqsQueueCommand.QueueUrl
    const attribsResponse = await sqsClient.send(new
 GetQueueAttributesCommand({QueueUrl: sqsQueueUrl, AttributeNames:
 ['QueueArn']}))
    const attribs = attribsResponse.Attributes
    console.log(attribs)
    const queueArn = attribs.QueueArn
    // subscribe SQS queue to SNS topic
    const subscribed = await snsClient.send(new SubscribeCommand({TopicArn:
 topicArn, Protocol:'sqs', Endpoint: queueArn}))
    const policy = {
      Version: "2012-10-17",&TCX5-2025-waiver;
      Statement: [
        {
          Sid: "MyPolicy",
          Effect: "Allow",
          Principal: {AWS: "*"},
          Action: "SQS:SendMessage",
          Resource: queueArn,
          Condition: {
            ArnEquals: {
              'aws:SourceArn': topicArn
            }
```

```
          }
        }
      ]
    };

    const response = sqsClient.send(new SetQueueAttributesCommand({QueueUrl:
 sqsQueueUrl, Attributes: {Policy: JSON.stringify(policy)}}))
    console.log(response)
    console.log(sqsQueueUrl, topicArn)
    return [sqsQueueUrl, topicArn]

  } catch (err) {
    console.log("Error", err);
  }
};

const startCelebrityDetection = async(roleArn, snsTopicArn) =>{
  try {
      //Initiate label detection and update value of startJobId with returned
 Job ID
      const response = await rekClient.send(new
 StartCelebrityRecognitionCommand({Video:{S3Object:{Bucket:bucket,
 Name:videoName}},
          NotificationChannel:{RoleArn: roleArn, SNSTopicArn: snsTopicArn}}))
          startJobId = response.JobId
          console.log(`Start Job ID: ${startJobId}`)
          return startJobId
    } catch (err) {
      console.log("Error", err);
    }
  };

const getCelebrityRecognitionResults = async(startJobId) =>{
  try {
      //Initiate label detection and update value of startJobId with returned
 Job ID
      var maxResults = 10
      var paginationToken = ''
      var finished = false

      while (finished == false){
          var response = await rekClient.send(new
 GetCelebrityRecognitionCommand({JobId: startJobId, MaxResults: maxResults,
              NextToken: paginationToken}))
```

```
            console.log(response.VideoMetadata.Codec)
            console.log(response.VideoMetadata.DurationMillis)
            console.log(response.VideoMetadata.Format)
            console.log(response.VideoMetadata.FrameRate)
            response.Celebrities.forEach(celebrityRecognition => {
                console.log(`Celebrity: ${celebrityRecognition.Celebrity.Name}`)
                console.log(`Timestamp: ${celebrityRecognition.Timestamp}`)
                console.log()
            })
            // Searh for pagination token, if found, set variable to next token
            if (String(response).includes("NextToken")){
                paginationToken = response.NextToken

            }else{
                finished = true
            }
        }
      } catch (err) {
        console.log("Error", err);
      }
    };

// Checks for status of job completion
const getSQSMessageSuccess = async(sqsQueueUrl, startJobId) => {
  try {
    // Set job found and success status to false initially
    var jobFound = false
    var succeeded = false
    var dotLine = 0
    // while not found, continue to poll for response
    while (jobFound == false){
      var sqsReceivedResponse = await sqsClient.send(new
 ReceiveMessageCommand({QueueUrl:sqsQueueUrl,
        MaxNumberOfMessages:'ALL', MaxNumberOfMessages:10}));
      if (sqsReceivedResponse){
        var responseString = JSON.stringify(sqsReceivedResponse)
        if (!responseString.includes('Body')){
          if (dotLine < 40) {
            console.log('.')
            dotLine = dotLine + 1
          }else {
            console.log('')
            dotLine = 0
          };
```

```
          stdout.write('', () => {
            console.log('');
          });
          await new Promise(resolve => setTimeout(resolve, 5000));
          continue
        }
      }

      // Once job found, log Job ID and return true if status is succeeded
      for (var message of sqsReceivedResponse.Messages){
        console.log("Retrieved messages:")
        var notification = JSON.parse(message.Body)
        var rekMessage = JSON.parse(notification.Message)
        var messageJobId = rekMessage.JobId
        if (String(rekMessage.JobId).includes(String(startJobId))){
          console.log('Matching job found:')
          console.log(rekMessage.JobId)
          jobFound = true
          console.log(rekMessage.Status)
          if (String(rekMessage.Status).includes(String("SUCCEEDED"))){
            succeeded = true
            console.log("Job processing succeeded.")
            var sqsDeleteMessage = await sqsClient.send(new
 DeleteMessageCommand({QueueUrl:sqsQueueUrl,
 ReceiptHandle:message.ReceiptHandle}));
          }
        }else{
          console.log("Provided Job ID did not match returned ID.")
          var sqsDeleteMessage = await sqsClient.send(new
 DeleteMessageCommand({QueueUrl:sqsQueueUrl,
 ReceiptHandle:message.ReceiptHandle}));
        }
      }
    }
  return succeeded
  } catch(err) {
    console.log("Error", err);
  }
};

// Start label detection job, sent status notification, check for success
 status
// Retrieve results if status is "SUCEEDED", delete notification queue and
 topic
```

```
const runCelebRecognitionAndGetResults = async () => {
  try {
    const sqsAndTopic = await createTopicandQueue();
    //const startLabelDetectionRes = await startLabelDetection(roleArn,
 sqsAndTopic[1]);
    //const getSQSMessageStatus = await getSQSMessageSuccess(sqsAndTopic[0],
 startLabelDetectionRes)
    const startCelebrityDetectionRes = await startCelebrityDetection(roleArn,
 sqsAndTopic[1]);
    const getSQSMessageStatus = await getSQSMessageSuccess(sqsAndTopic[0],
 startCelebrityDetectionRes)
    console.log(getSQSMessageSuccess)
    if (getSQSMessageSuccess){
      console.log("Retrieving results:")
      const results = await
 getCelebrityRecognitionResults(startCelebrityDetectionRes)
    }
    const deleteQueue = await sqsClient.send(new DeleteQueueCommand({QueueUrl:
 sqsAndTopic[0]}));
    const deleteTopic = await snsClient.send(new DeleteTopicCommand({TopicArn:
 sqsAndTopic[1]}));
    console.log("Successfully deleted.")
  } catch (err) {
    console.log("Error", err);
  }
};

runCelebRecognitionAndGetResults()
```

CLI

Ejecute el siguiente AWS CLI comando para empezar a detectar celebridades en un vídeo.

```
aws rekognition start-celebrity-recognition --video "{"S3Object":
{"Bucket":"amzn-s3-demo-bucket","Name":"video-name"}}" \
--notification-channel "{"SNSTopicArn":"topic-arn","RoleArn":"role-arn"}" \
--region region-name --profile profile-name
```

Actualice los siguientes valores:

- Cambie `amzn-s3-demo-bucket` y `video-name` por el nombre del bucket de Amazon S3 y el nombre de archivo que especificó en el paso 2.

- Cambie `region-name` por la región de AWS que está utilizando.
- Sustituya el valor de `profile-name` de por el nombre de su perfil de desarrollador.
- Reemplace `topic-ARN` por el ARN del tema de Amazon SNS que creó en el paso 3 de Configuración de Amazon Rekognition Video.
- Cambie `role-ARN` por el ARN del rol de servicio de IAM que creó en el paso 7 de Configuración de Amazon Rekognition Video.

Si accede a la CLI en un dispositivo Windows, utilice comillas dobles en lugar de comillas simples y evite las comillas dobles internas con una barra invertida (es decir, \) para corregir cualquier error del analizador que pueda encontrar. Consulte a continuación un ejemplo:

```
aws rekognition start-celebrity-recognition --video "{\"S3Object\":{\"Bucket\":
\"amzn-s3-demo-bucket\",\"Name\":\"video-name\"}}" \
--notification-channel "{\"SNSTopicArn\":\"topic-arn\",\"RoleArn\":\"role-arn
\"}" \
--region region-name --profile profile-name
```

Tras ejecutar el ejemplo de código de procedimiento, copie el `jobID` devuelto y envíelo al siguiente comando `GetCelebrityRecognition` para obtener los resultados, sustituyendo `job-id-number` por el `jobID` que recibió anteriormente:

```
aws rekognition get-celebrity-recognition --job-id job-id-number --profile
 profile-name
```

> **Note**
>
> Si ya ha ejecutado un ejemplo de vídeo distinto de Análisis de un vídeo almacenado en un bucket de Amazon S3 con Java o Python (SDK), el código que se va a reemplazar podría ser diferente.

3. Ejecute el código. Se muestra información sobre los famosos reconocidos en el vídeo.

GetCelebrityRecognition respuesta de operación

A continuación, se muestra un ejemplo de respuesta JSON. La respuesta incluye lo siguiente:

- Famosos reconocidos: `Celebrities` es una matriz de famosos y las veces que se reconocen en un vídeo. Hay un objeto CelebrityRecognition por cada momento en que se reconoce al famoso en el vídeo. Cada `CelebrityRecognition` contiene información sobre un famoso reconocido (CelebrityDetail) y la hora (`Timestamp`) a la que se ha reconocido el famoso en el vídeo. `Timestamp` se mide en milisegundos desde el comienzo del vídeo.
- CelebrityDetail— Contiene información sobre una celebridad reconocida. Incluye el nombre de la celebridad (`Name`), el identificador (`ID`), el género conocido de la celebridad (`KnownGender`) y una lista que URLs apunta a contenido relacionado (`Urls`). También incluye el nivel de confianza que Amazon Rekognition Video tiene en la precisión del reconocimiento y detalles sobre el rostro de la celebridad. FaceDetail Si necesita obtener contenido relacionado más adelante, puede utilizar `ID` con getCelebrityInfo.
- VideoMetadata— Información sobre el vídeo que se analizó.


```
{
    "Celebrities": [
        {
            "Celebrity": {
                "Confidence": 0.699999988079071,
                "Face": {
                    "BoundingBox": {
                        "Height": 0.20555555820465088,
                        "Left": 0.029374999925494194,
                        "Top": 0.22333332896232605,
                        "Width": 0.11562500149011612
                    },
                    "Confidence": 99.89837646484375,
                    "Landmarks": [
                        {
                            "Type": "eyeLeft",
                            "X": 0.06857934594154358,
                            "Y": 0.30842265486717224
                        },
                        {
                            "Type": "eyeRight",
                            "X": 0.10396526008844376,
                            "Y": 0.300625205039978
```

```
                        },
                        {
                            "Type": "nose",
                            "X": 0.0966852456331253,
                            "Y": 0.34081998467445374
                        },
                        {
                            "Type": "mouthLeft",
                            "X": 0.075217105448246,
                            "Y": 0.3811396062374115
                        },
                        {
                            "Type": "mouthRight",
                            "X": 0.10744428634643555,
                            "Y": 0.37407416105270386
                        }
                    ],
                    "Pose": {
                        "Pitch": -0.9784082174301147,
                        "Roll": -8.808176040649414,
                        "Yaw": 20.28228759765625
                    },
                    "Quality": {
                        "Brightness": 43.312068939208984,
                        "Sharpness": 99.9305191040039
                    }
                },
                "Id": "XXXXXX",
                "KnownGender": {
                    "Type": "Female"
                },
                "Name": "Celeb A",
                "Urls": []
            },
            "Timestamp": 367
        },......
    ],
    "JobStatus": "SUCCEEDED",
    "NextToken": "XfXnZKiyMOGDhzBzYUhS5puM+g1IgezqFeYpv/H/+5noP/LmM57FitUAwSQ5D6G4AB/
PNwolrw==",
    "VideoMetadata": {
        "Codec": "h264",
        "DurationMillis": 67301,
        "FileExtension": "mp4",
```

```
        "Format": "QuickTime / MOV",
        "FrameHeight": 1080,
        "FrameRate": 29.970029830932617,
        "FrameWidth": 1920
    }
}
```

Obtención de información sobre un famoso

En estos procedimientos, obtendrá información de famosos utilizando la operación API getCelebrityInfo. El famoso se identifica mediante el ID de famoso que devuelve una llamada a anterior a RecognizeCelebrities.

¿Llamando GetCelebrityInfo

Estos procedimientos también requieren el ID de famoso para un famoso que Amazon Rekognition conoce. Utilice el ID de famoso que anotó en Reconocimiento de famosos en una imagen.

Para obtener información sobre un famoso (SDK)

1. Si aún no lo ha hecho:

   a. Cree o actualice un usuario con los permisos `AmazonRekognitionFullAccess` y `AmazonS3ReadOnlyAccess`. Para obtener más información, consulte Paso 1: Configurar una cuenta de AWS y crear un usuario.

   b. Instale y configure el AWS CLI y AWS SDKs. Para obtener más información, consulte Paso 2: Configure y AWS CLI AWS SDKs.

2. Consulte los siguientes ejemplos para llamar a la operación `GetCelebrityInfo`.

   Java

   Este ejemplo muestra el nombre y la información sobre un famoso.

   `id`Sustitúyalo por uno de los famosos IDs que aparecen enReconocimiento de famosos en una imagen.

```
//Copyright 2018 Amazon.com, Inc. or its affiliates. All Rights Reserved.
//PDX-License-Identifier: MIT-0 (For details, see https://github.com/awsdocs/
amazon-rekognition-developer-guide/blob/master/LICENSE-SAMPLECODE.)
```

```
package aws.example.rekognition.image;
import com.amazonaws.services.rekognition.AmazonRekognition;
import com.amazonaws.services.rekognition.AmazonRekognitionClientBuilder;
import com.amazonaws.services.rekognition.model.GetCelebrityInfoRequest;
import com.amazonaws.services.rekognition.model.GetCelebrityInfoResult;


public class CelebrityInfo {

   public static void main(String[] args) {
      String id = "nnnnnnnn";

      AmazonRekognition rekognitionClient =
 AmazonRekognitionClientBuilder.defaultClient();

      GetCelebrityInfoRequest request = new GetCelebrityInfoRequest()
         .withId(id);

      System.out.println("Getting information for celebrity: " + id);

      GetCelebrityInfoResult
 result=rekognitionClient.getCelebrityInfo(request);

      //Display celebrity information
      System.out.println("celebrity name: " + result.getName());
      System.out.println("Further information (if available):");
      for (String url: result.getUrls()){
         System.out.println(url);
      }
   }
}
```

Java V2

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import
 software.amazon.awssdk.services.rekognition.model.GetCelebrityInfoRequest;
```

```
import
 software.amazon.awssdk.services.rekognition.model.GetCelebrityInfoResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class CelebrityInfo {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <id>

                Where:
                   id - The id value of the celebrity. You can use the
 RecognizeCelebrities example to get the ID value.\s
                """;

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

        String id = args[0];
        Region region = Region.US_WEST_2;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        getCelebrityInfo(rekClient, id);
        rekClient.close();
    }

    /**
     * Retrieves information about a celebrity identified in an image.
     *
     * @param rekClient the Amazon Rekognition client used to make the API call
     * @param id the unique identifier of the celebrity
```

```
     * @throws RekognitionException if there is an error retrieving the
 celebrity information
     */
    public static void getCelebrityInfo(RekognitionClient rekClient, String id)
 {
        try {
            GetCelebrityInfoRequest info = GetCelebrityInfoRequest.builder()
                    .id(id)
                    .build();

            GetCelebrityInfoResponse response =
 rekClient.getCelebrityInfo(info);
            System.out.println("celebrity name: " + response.name());
            System.out.println("Further information (if available):");
            for (String url : response.urls()) {
                System.out.println(url);
            }

        } catch (RekognitionException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

AWS CLI

Este AWS CLI comando muestra el resultado JSON de la operación `get-celebrity-info` CLI. `ID`Sustitúyala por una de las celebridades IDs que aparecen en[Reconocimiento de famosos en una imagen](). Sustituya el valor de `profile-name` de por el nombre de su perfil de desarrollador.

```
aws rekognition get-celebrity-info --id celebrity-id --profile profile-name
```

Python

Este ejemplo muestra el nombre y la información sobre un famoso.

`id`Sustitúyala por una de las celebridades IDs que aparecen en[Reconocimiento de famosos en una imagen](). Sustituya el valor de `profile_name` en la línea que crea la sesión de Rekognition por el nombre de su perfil de desarrollador.

```
# Copyright 2018 Amazon.com, Inc. or its affiliates. All Rights Reserved.
# PDX-License-Identifier: MIT-0 (For details, see https://github.com/awsdocs/
amazon-rekognition-developer-guide/blob/master/LICENSE-SAMPLECODE.)

import boto3

def get_celebrity_info(id):

    session = boto3.Session(profile_name='profile-name')
    client = session.client('rekognition')

    # Display celebrity info
    print('Getting celebrity info for celebrity: ' + id)

    response = client.get_celebrity_info(Id=id)

    print(response['Name'])
    print('Further information (if available):')
    for url in response['Urls']:
        print(url)

def main():
    id = "celebrity-id"
    celebrity_info = get_celebrity_info(id)

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

.NET

Este ejemplo muestra el nombre y la información sobre un famoso.

`id`Sustitúyala por una de las celebridades IDs que aparecen en[Reconocimiento de famosos en una imagen](#).

```
//Copyright 2018 Amazon.com, Inc. or its affiliates. All Rights Reserved.
//PDX-License-Identifier: MIT-0 (For details, see https://github.com/awsdocs/
amazon-rekognition-developer-guide/blob/master/LICENSE-SAMPLECODE.)

using System;
using Amazon.Rekognition;
using Amazon.Rekognition.Model;
```

```
public class CelebrityInfo
{
    public static void Example()
    {
        String id = "nnnnnnnn";

        AmazonRekognitionClient rekognitionClient = new
 AmazonRekognitionClient();

        GetCelebrityInfoRequest celebrityInfoRequest = new
 GetCelebrityInfoRequest()
        {
            Id = id
        };

        Console.WriteLine("Getting information for celebrity: " + id);

        GetCelebrityInfoResponse celebrityInfoResponse =
 rekognitionClient.GetCelebrityInfo(celebrityInfoRequest);

        //Display celebrity information
        Console.WriteLine("celebrity name: " + celebrityInfoResponse.Name);
        Console.WriteLine("Further information (if available):");
        foreach (String url in celebrityInfoResponse.Urls)
            Console.WriteLine(url);
    }
}
```

GetCelebrityInfo solicitud de operación

A continuación se ofrece un ejemplo de JSON de la salida y la entrada de `GetCelebrityInfo`.

La entrada de `GetCelebrityInfo` es el ID del famoso requerido.

```
{
    "Id": "nnnnnnn"
}
```

GetCelebrityInfo respuesta de operación

`GetCelebrityInfo` devuelve una matriz (`Urls`) de enlaces a la información sobre el famoso solicitado.

```
{
    "Name": "Celebrity Name",
    "Urls": [
        "www.imdb.com/name/nmnnnnnnn"
    ]
}
```

Moderación del contenido

Puede usar Amazon Rekognition para detectar contenido inapropiado, no deseado u ofensivo. Puede utilizar la APIs moderación de Rekognition en las redes sociales, los medios de difusión, la publicidad y el comercio electrónico para crear una experiencia de usuario más segura, ofrecer garantías de seguridad de la marca a los anunciantes y cumplir con las normativas locales y globales.

Hoy en día, muchas empresas confían exclusivamente en moderadores humanos para revisar el contenido generado por terceros o por los usuarios, mientras que otras simplemente reaccionan ante las quejas de los usuarios para eliminar imágenes, anuncios o vídeos ofensivos o inapropiados. Sin embargo, los moderadores humanos por sí solos no pueden escalarse para satisfacer estas necesidades con la calidad o la velocidad suficientes, lo que se traduce en una mala experiencia de usuario, en unos costes elevados para escalar o incluso en la pérdida de reputación de la marca. Al utilizar Rekognition para la moderación de imágenes y vídeos, los moderadores humanos pueden revisar un conjunto de contenido mucho más reducido, normalmente del 1 al 5 % del volumen total, que ya ha sido marcado por machine learning. Esto les permite centrarse en actividades más valiosas y, aun así, conseguir una cobertura integral de moderación por una fracción de su coste actual. Para configurar una fuerza laboral humana y realizar tareas de revisión humana, puede usar Amazon Augmented AI, que está integrada con Rekognition.

Puede mejorar la precisión del modelo de aprendizaje profundo de moderación con la característica de moderación personalizada. Con la moderación personalizada, puede entrenar un adaptador de moderación personalizado cargando sus imágenes y anotándolas. Luego, se puede proporcionar el adaptador adecuado a la DetectModerationLabelsoperación para mejorar su rendimiento en las imágenes. Para obtener más información, consulte Mejora de la precisión con la moderación personalizada.

Etiquetas compatibles con las operaciones de moderación de contenido de Rekognition

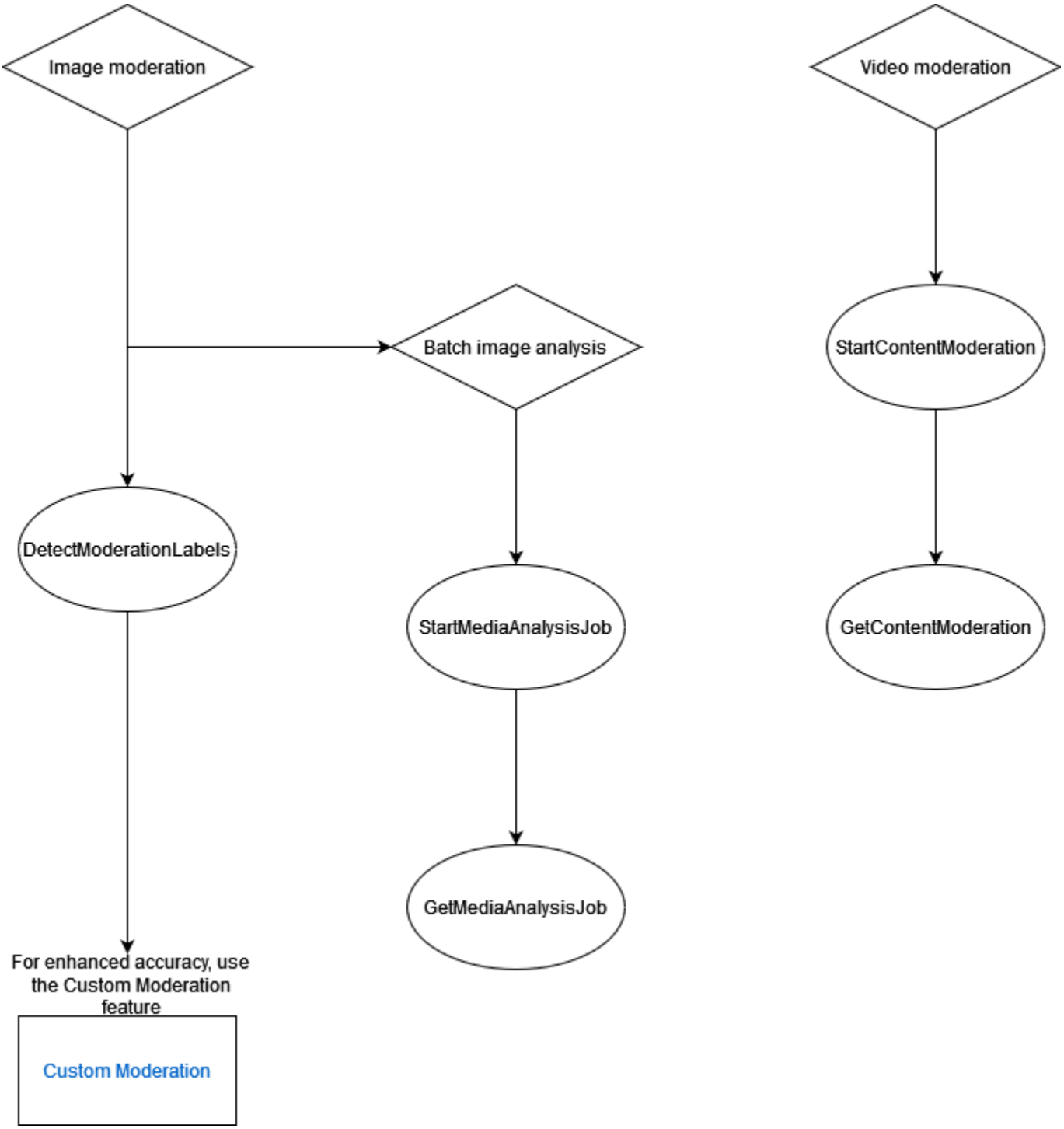
- Para descargar una lista de las etiquetas de moderación, haga clic aquí.

Temas

- Uso de la moderación de imágenes y vídeos APIs
- Cómo probar la versión 7 de Moderación de contenido y transformar la respuesta de la API
- Detección de imágenes inapropiadas
- Detectar vídeos almacenados inapropiados

- Mejora de la precisión con la moderación personalizada
- Revisión de contenido inapropiado con Amazon Augmented AI

En el siguiente diagrama se muestra el orden de las operaciones de llamada, en función de sus objetivos de uso de los componentes de imagen o vídeo de Moderación de contenido:



Uso de la moderación de imágenes y vídeos APIs

En la API Amazon Rekognition Image, puede detectar contenido inapropiado, no deseado u ofensivo mediante el uso y las operaciones de DetectModerationLabelsforma sincrónica y asíncrona. StartMediaAnalysisJobGetMediaAnalysisJob Puede usar la API Amazon Rekognition Video para detectar dicho contenido de forma asíncrona mediante las operaciones y. StartContentModerationGetContentModeration

Categorías de etiquetas

Amazon Rekognition utiliza una taxonomía jerárquica de tres niveles para etiquetar las categorías de contenido inapropiado, no deseado u ofensivo. Cada etiqueta con el nivel de taxonomía (L1) tiene varias etiquetas del nivel de taxonomía 2 (L2) y algunas etiquetas del nivel de taxonomía 2 pueden tener etiquetas del nivel de taxonomía 3 (L3). Esto permite una clasificación jerárquica del contenido.

Para cada etiqueta de moderación detectada, la API también devuelve el valor `TaxonomyLevel`, que contiene el nivel (1, 2 o 3) al que pertenece la etiqueta. Por ejemplo, una imagen puede etiquetarse de acuerdo con la siguiente categorización:

L1: desnudez no explícita de partes íntimas y besos; L2: desnudez no explícita; L3: desnudez implícita.

> **Note**
>
> Se recomienda usar las categorías L1 o L2 para moderar el contenido y utilizar las categorías L3 solo para eliminar conceptos específicos que no quiera moderar (es decir, para detectar contenido que puede que no quiera categorizar como inapropiado, no deseado u ofensivo en función de su política de moderación).

En la siguiente tabla se muestran las relaciones entre los niveles de las categorías y las posibles etiquetas para cada nivel. Para descargar una lista de las etiquetas de moderación, haga clic aquí.

Categoría de nivel superior (L1)	Categoría de segundo nivel (L2)	Categoría de tercer nivel (L3)	Definiciones
Explícita	Desnudo explícito	Genitales masculinos expuestos	Los genitales masculinos humanos,

		incluidos el pene (erecto o flácido), el escroto y cualquier vello púbico discernible. Este término se aplica en contextos que implican actividad sexual o cualquier contenido visual en el que los genitales masculinos se muestren total o parcialmente.
	Genitales femeninos expuestos	Partes externas del aparato reproductor femenino, que incluyen la vulva, la vagina y cualquier vello púbico observable. Este término es aplicable en escenarios que implican actividad sexual o cualquier contenido visual en el que estos aspectos de la anatomía femenina se muestran total o parcialmente.

	Glúteos o ano expuestos	Los glúteos o el ano humanos, incluidos los casos en los que los glúteos están desnudos o cuando se pueden discernir a través de ropa transparente. La definición se aplica específicamente a las situaciones en las que las nalgas o el ano son visibles de forma directa y completa, con exclusión de los casos en los que cualquier tipo de ropa interior o ropa proporciona una cobertura total o parcial.
	Pezón femenino expuesto	Los pezones de las mujeres humanas, incluida la aerola (zona que rodea los pezones) y los pezones, totalmente visibles y parcialme nte visibles.

Actividad sexual explícita	N/A	Representación de actos sexuales reales o simulados que abarcan las relaciones sexuales humanas, el sexo oral, así como la estimulación genital masculina y la estimulación genital femenina mediante otras partes y objetos del cuerpo. El término también incluye la eyaculación o los fluidos vaginales en partes del cuerpo y las prácticas eróticas o los juegos de rol relacionados con la esclavitud, la disciplina, el dominio y la sumisión, y el sadomasoquismo.
Juguetes sexuales	N/A	Objetos o dispositivos utilizados para la estimulación o el placer sexual, por ejemplo, un consolador, un vibrador, un tapón anal, ritmos, etc.

Desnudos no explícitos de partes íntimas y besos	Desnudez no explícita	Espalda desnuda	Parte posterior humana donde se ve la mayor parte de la piel desde el cuello hasta el final de la columna vertebral. Este término no se aplica cuando la espalda de la persona está parcial o totalmente ocluida.
		Pezón masculino expuesto	Pezones masculinos humanos, incluidos los pezones parcialmente visibles.
		Glúteos parcialmente expuestos	Glúteos humanos parcialmente expuestos. Este término incluye una región parcialmente visible de las nalgas o las nalgas debido a la ropa corta, o una parte superior parcialmente visible de la hendidura anal. El término no se aplica a los casos en los que las nalgas están completamente desnudas.

		Mama femenina parcialmente expuesta	Seno femenino humano parcialmente expuesto, donde una parte del seno de la mujer está visible o descubierta sin dejar al descubierto todo el seno. Este término se aplica cuando se ve la región del pliegue interno de la mama o cuando se ve el pliegue inferior de la mama con el pezón completamente cubierto u ocluido.
		Desnudez implícita	Una persona que está desnuda, ya sea en topless o sin fondo, pero con partes íntimas como las nalgas, los pezones o los genitales cubiertas, ocluidas o no completamente visibles.
	Partes íntimas obstruidas	Pezón femenino obstruido	Representación visual de una situación en la que los pezones de una mujer están cubiertos por ropa o coberturas opacas, pero sus formas son claramente visibles.

		Genitales masculinos obstruidos	Representación visual de una situación en la que los genitales o el pene de un hombre están cubiertos por ropa o coberturas opacas, pero su forma es claramente visible. Este término se aplica cuando los genitales obstruidos de la imagen aparecen en primer plano.
	Besos en los labios	N/A	Representación de los labios de una persona haciendo contacto con los labios de otra persona.
Trajes de baño o ropa interior	Trajes de baño o ropa interior femenina	N/A	Ropa humana para trajes de baño femeninos (por ejemplo, trajes de baño de una sola pieza, bikinis, tankinis, etc.) y ropa interior femenina (por ejemplo, sujetadores, bragas, calzoncillos, lencería, tangas, etc.)

	Trajes de baño o ropa interior masculinos	N/A	Ropa humana para trajes de baño masculinos (por ejemplo, bañadores, bañadores, calzoncillos, etc.) y ropa interior masculina (por ejemplo, calzoncillos, bóxers, etc.)
Violencia	Armas	N/A	Instrumentos o dispositivos utilizados para causar daños o lesiones a seres vivos, estructuras o sistemas. Esto incluye armas de fuego (por ejemplo, pistolas, fusiles, ametralladoras, etc.), armas afiladas (por ejemplo, espadas, cuchillos, etc.), explosivos y municiones (por ejemplo, misiles, bombas, balas, etc.).
	Violencia gráfica	Violencia con armas	El uso de armas para causar daños, lesiones o la muerte a uno mismo, a otras personas o a propiedades.

Violencia física	El acto de causar daño a otras personas o bienes (por ejemplo, golpear, pelear, arrancarles el pelo, etc.) u otro acto de violencia que involucre a una multitud o a varias personas.
Autolesión	El acto de hacerse daño a uno mismo, a menudo cortándose partes del cuerpo, como brazos o piernas, donde los cortes suelen ser visibles.
Sangre y sangre	Representación visual de la violencia ejercida sobre una persona, un grupo de personas o un animal, con heridas abiertas, derramamiento de sangre y partes del cuerpo mutiladas.
Explosiones y explosiones	Representación de un violento y destructivo estallido de llamas intensas con humo espeso o polvo y humo que sale del suelo.

Visualmente perturbador	Muerte y emaciación	Cuerpos desnutridos	Cuerpos humanos extremadamente delgados y desnutridos, con una grave atrofia física y agotamiento del tejido muscular y adiposo.
		Cadáveres	Cadáveres humanos en forma de cuerpos mutilados, cadáveres colgados o esqueletos.
	Se estrella	Accidentes aéreos	Incidentes de vehículos aéreos, como aviones, helicópteros u otros vehículos voladores, que provoquen daños, lesiones o la muerte. Este término se aplica cuando se ven partes de los vehículos aéreos.

Drogas y tabaco	Productos	Pastillas	Mesas o cápsulas pequeñas, sólidas, a menudo redondas u ovaladas. Este término se aplica a las píldoras que se presentan solas, en un frasco o en un paquete transparente y no se aplica a una representación visual de una persona que toma las píldoras.
	Parafernalia y consumo de drogas y tabaco	Fumar	El acto de inhalar, exhalar y encender sustancias que se queman, como cigarrillos, puros, cigarrillos electróni cos, narguiles o porros.
Alcohol	Consumo de alcohol	Beber	El acto de beber bebidas alcohólicas en botellas o vasos de alcohol o licor.

	Bebidas alcohólicas	N/A	Primer plano de una o varias botellas de alcohol o licor, vasos o jarras con alcohol o licor y vasos o jarras con alcohol o licor en poder de una persona. Este término no se aplica a una persona que beba de botellas o vasos de alcohol o licor.
Gestos groseros	Cortes de manga	N/A	Representación visual de un gesto con la mano con el dedo medio extendido hacia arriba mientras los otros dedos están doblados hacia abajo.
Apuestas	N/A	N/A	El acto de participa r en juegos de azar para tener la oportunidad de ganar un premio en los casinos, por ejemplo, jugar a las cartas, al blackjack, a la ruleta, a las máquinas tragaperras de los casinos, etc.

Símbolos de odio	Partido Nazi	N/A	Representación visual de símbolos, banderas o gestos relacionados con el Partido Nazi.
	Supremacía blanca	N/A	Representación visual de símbolos o prendas relacionados con el Ku Klux Klan (KKK) e imágenes con banderas de la Confederación.
	Extremismo	N/A	Imágenes que contienen banderas de grupos extremistas y terroristas.

No todas las etiquetas de la categoría L2 tienen una etiqueta compatible en la categoría L3. Además, las etiquetas L3 bajo las etiquetas L2 “Productos” y “Accesorios y consumo de drogas y tabaco” no son exhaustivas. Estas etiquetas L2 abarcan conceptos que van más allá de las etiquetas L3 mencionadas y, en esos casos, solo se devuelven las etiquetas L2 en la respuesta de la API.

Puede determinar la idoneidad para su aplicación. Por ejemplo, es posible que las imágenes de naturaleza insinuante sean aceptables, pero no lo sean las imágenes que contengan desnudos. Para filtrar imágenes, utilice la matriz de etiquetas ModerationLabel devuelta por `DetectModerationLabels` (imágenes) y por `GetContentModeration` (vídeos).

Tipo de contenido

La API también puede identificar si el tipo de contenido es animado o ilustrado, lo que se devuelve como parte de la respuesta:

- El contenido animado incluye videojuegos y animaciones (por ejemplo, dibujos animados, cómics, manga o anime).
- El contenido ilustrado incluye dibujos, pinturas y bocetos.

Confianza

Puede configurar el umbral de confianza que Amazon Rekognition utiliza para detectar contenido inapropiado especificando el parámetro de entrada `MinConfidence`. No se devuelven etiquetas de contenido inapropiado que se detectan con una confianza menor al valor de `MinConfidence`.

Si se especifica un valor de `MinConfidence` inferior a un 50 %, es probable que se devuelvan resultados con un número elevado de falsos positivos (es decir, mayor recuperación, menor precisión). Por otro lado, si se especifica un valor de `MinConfidence` por encima del 50 %, es probable que se devuelvan resultados con un número menor de falsos positivos (es decir, menor recuperación, mayor precisión). Si no especifica ningún valor para `MinConfidence`, Amazon Rekognition devuelve etiquetas para el contenido inapropiado detectado con una confianza de al menos el 50 %.

La matriz `ModerationLabel` contiene etiquetas de las categorías anteriores y la confianza estimada en la precisión del contenido reconocido. Se devuelve una etiqueta de nivel superior junto con todas las etiquetas de segundo nivel identificadas. Por ejemplo, Amazon Rekognition puede devolver «Desnudo explícito» con una puntuación de confianza alta como etiqueta de nivel superior. Esto podría ser suficiente para satisfacer sus necesidades de filtrado. Sin embargo, si es necesario, puede utilizar la puntuación de confianza de una etiqueta de segundo nivel (como "Desnudo masculino gráfico") para obtener un filtrado más detallado. Para ver un ejemplo, consulta Detección de imágenes inapropiadas.

Control de versiones

Tanto Amazon Rekognition Image como Amazon Rekognition Video devuelven la versión del modelo de detección de moderación que se utiliza para detectar contenido inapropiado (`ModerationModelVersion`).

Clasificación y agregación

Al recuperar los resultados con GetContentModeration, puede ordenarlos y agregarlos.

Orden de clasificación: la matriz de etiquetas devueltas está ordenada por tiempo. Para ordenar por etiqueta, especifique `NAME` en el parámetro de entrada `SortBy` para `GetContentModeration`. Si la etiqueta aparece varias veces en el vídeo, habrá varias instancias del elemento `ModerationLabel`.

Información de etiqueta: el elemento de ModerationLabels matriz contiene un `ModerationLabel` objeto que, a su vez, contiene el nombre de la etiqueta y la confianza que Amazon Rekognition tiene

en la precisión de la etiqueta detectada. La marca de tiempo es la hora en que `ModerationLabel` se detectó, definida como el número de milisegundos transcurridos desde el inicio del vídeo. Para los resultados agregados por vídeo `SEGMENTS`, se devuelven las estructuras `StartTimestampMillis`, `EndTimestampMillis` y `DurationMillis`, que definen la hora de inicio, la hora de finalización y la duración de un segmento, respectivamente.

Agregación: especifica cómo se agregan los resultados cuando se devuelven. El valor predeterminado es agregar por `TIMESTAMPS`. También puede optar por agregar por `SEGMENTS`, lo que agrega los resultados en un intervalo de tiempo. Solo se devuelven las etiquetas detectadas durante los segmentos.

Estados de adaptador de Moderación personalizada

Los adaptadores de Moderación personalizada pueden tener uno de los siguientes estados: TRAINING_IN_PROGRESS, TRAINING_COMPLETED, TRAINING_FAILED, DELETING, DEPRECATED o EXPIRED. Para obtener una explicación completa de estos estados de los adaptadores, consulte Administración de adaptadores.

> **Note**
>
> Amazon Rekognition no es una autoridad en la materia ni pretende en modo alguno ser un filtro exhaustivo de contenido ofensivo o inapropiado. Además, la moderación de imágenes y vídeos APIs no detecta si una imagen incluye contenido ilegal, como el CSAM.

Cómo probar la versión 7 de Moderación de contenido y transformar la respuesta de la API

Rekognition ha actualizado el modelo de machine learning para los componentes de imagen y vídeo de la característica de detección de etiquetas de Moderación de contenido de la versión 6.1 a la 7. Esta actualización ha mejorado la precisión general y ha introducido varias categorías nuevas, además de modificar otras.

Si actualmente es usuario de vídeo de la versión 6.1, le recomendamos que tome las siguientes medidas para realizar la transición a la versión 7 sin problemas:

1. Descargue y utilice un SDK privado de AWS (consulte lathe section called “AWS SDK y guía de uso para la moderación de contenido, versión 7”) para llamar a la StartContentModeration API.

2. Revise la lista actualizada de etiquetas y puntuaciones de confianza que devuelve la consola o la respuesta de la API. Si es necesario, ajuste la lógica de posprocesamiento de la aplicación en consecuencia.
3. Su cuenta conservará la versión 6.1 hasta el 13 de mayo de 2024. Si quiere utilizar la versión 6.1 después del 13 de mayo de 2024, póngase en contacto con el equipo de AWS Support antes del 30 de abril de 2024 para solicitar una extensión. Podemos ampliar la permanencia de su cuenta en la versión 6.1 hasta el 10 de junio de 2024. Si no recibimos noticias suyas antes del 30 de abril de 2024, la cuenta se migrará de forma automática a la versión 7.0 a partir del 13 de mayo de 2024.

AWS SDK y guía de uso para la moderación de contenido, versión 7

Descargue el SDK que corresponda al lenguaje de desarrollo que haya elegido y consulte la guía de usuario correspondiente.

Enlace a SDK	Guía de usuario/instalación
Java-1.X	Guía: Java 1.pdf
Java-2.X	Guía: Java 2.pdf
JavaScript v2	Guía: JavaScript v2.pdf
JavaScript v3	Guía: JavaScript v3.pdf
Python	Guía: Python y AWS CLI.pdf
Ruby	Guía: RubyV3.pdf
go_v1	Guía: GO V1.pdf
go_v2	Guía: GO V2.pdf
DotNet	Guía: .net.pdf
php	Guía: PHP.pdf

Asignaciones de etiquetas para las versiones 6.1 a 7

La versión 7 de Moderación de contenido ha agregado nuevas categorías de etiquetas y ha modificado nombres de etiquetas ya existentes. Consulte la tabla de taxonomía que se encuentra en the section called “Categorías de etiquetas” a la hora de decidir cómo asignar etiquetas de la versión 6.1 a etiquetas de la 7.

En la siguiente sección hay algunos ejemplos de asignaciones de etiquetas. Se recomienda que revise estas asignaciones y las definiciones de las etiquetas antes de realizar las actualizaciones necesarias en función de la lógica de posprocesamiento de la aplicación.

Esquema de asignaciones de L1

Si usa una lógica de posprocesamiento que solo filtra la categoría de nivel superior (L1) (por ejemplo, `Explicit Nudity`, `Suggestive`, `Violence`, etc.), consulte la tabla siguiente para actualizar el código.

L1 V6.1	L1 V7
Desnudo explícito	Explícita
Insinuante	Desnudez no explícita de partes íntimas y besos
	Trajes de baño o ropa interior
Violencia	Violencia
Visualmente perturbador	Visualmente perturbador
Gestos groseros	Gestos groseros
Drogas	Drogas y tabaco
Tabaco	Drogas y tabaco
Alcohol	Alcohol
Apuestas	Apuestas
Símbolos de odio	Símbolos de odio

Esquema de asignaciones de L2

Si usa una lógica de posprocesamiento que filtra las categorías L1 y L2 (por ejemplo, `Explicit Nudity / Nudity, Suggestive / Female Swimwear Or Underwear`, `Violence / Weapon Violence`, etc.), consulte la tabla siguiente para actualizar el código.

L1 V6.1	L2 V6.1	L1 V7	L2 V7	L3 V7	V7 ContentTypes
Desnudo explícito	Desnudo	Explícita	Desnudo explícito	Pezón femenino expuesto	
				Glúteos o ano expuestos	
	Desnudo masculina gráfica	Explícita	Desnudo explícito	Genitales masculinos expuestos	
	Desnudo femenino gráfico	Explícita	Desnudo explícito	Genitales femeninos expuestos	
	Actividad sexual	Explícita	Actividad sexual explícita		
	Desnudez explícita ilustrada	Explícita	Desnudo explícito		Mapa para «Animado» e «Ilustrado»
	Desnudez explícita ilustrada	Explícita	Actividad sexual explícita		Mapa para «Animado» e «Ilustrado»
	Juguetes para adultos	Explícita	Juguetes sexuales		

Insinuante	Roba de baño o ropa interior femenina	Trajes de baño o ropa interior	Trajes de baño o ropa interior femenina	
	Roba de baño o ropa interior masculina	Trajes de baño o ropa interior	Trajes de baño o ropa interior masculinos	
	Desnudo parcial	Desnudez no explícita de partes íntimas y besos	Desnudez no explícita	Desnudez implícita
	Hombre con el pecho descubierto	Desnudez no explícita de partes íntimas y besos	Desnudez no explícita	Pezón masculino expuesto
	Ropa provocativa	Desnudez no explícita de partes íntimas y besos	Desnudez no explícita	
		Desnudez no explícita de partes íntimas y besos	Partes íntimas obstruidas	

	Situaciones sexuales	Desnudez no explícita de partes íntimas y besos	Besos en los labios	
Violencia	Violencia gráfica o sangre	Violencia	Violencia gráfica	Sangre y sangre
	Violencia física	Violencia	Violencia gráfica	Violencia física
	Violencia con armas	Violencia	Violencia gráfica	Violencia con armas
	Armas	Violencia	Armas	
	Autolesiones	Violencia	Violencia gráfica	Autolesión
Visualmente perturbador	Cuerpos desnutridos	Visualmente perturbador	Muerte y emaciación	Cuerpos desnutridos
	Cadáveres	Visualmente perturbador	Muerte y emaciación	Cadáveres
	Personas colgadas	Visualmente perturbador	Muerte y emaciación	Cadáveres
	Accidentes aéreos	Visualmente perturbador	Choques	Accidentes aéreos
	Explosiones y deflagraciones	Violencia	Violencia gráfica	Explosiones y explosiones
Gestos groseros	Cortes de manga	Gestos groseros	Cortes de manga	

Drogas	Productos farmacéuticos	Drogas y tabaco	Productos	
	Uso de drogas	Drogas y tabaco	Paraferna lia y uso de drogas y tabaco	
	Pastillas	Drogas y tabaco	Productos	Pastillas
	Paraferna lia relaciona da con las drogas	Drogas y tabaco	Paraferna lia y uso de drogas y tabaco	
Tabaco	Productos de tabaco	Drogas y tabaco	Productos	
	Fumar	Drogas y tabaco	Paraferna lia y uso de drogas y tabaco	Fumar
Alcohol	Beber	Alcohol	Consumo de alcohol	Beber
	Bebidas alcohólicas	Alcohol	Bebidas alcohólicas	
Apuestas	Apuestas	Apuestas		
Símbolos de odio	Partido Nazi	Símbolos de odio	Partido Nazi	
	Supremacía blanca	Símbolos de odio	Supremacía blanca	

Extremismo	Símbolos de odio	Extremismo

Detección de imágenes inapropiadas

Puede utilizar la DetectModerationLabelsoperación para determinar si una imagen contiene contenido inapropiado u ofensivo. Para ver una lista de las etiquetas de moderación en Amazon Rekognition, consulte Uso de la moderación de imágenes y vídeos. APIs

Detección de contenido inapropiado en una imagen

La imagen debe estar en formato .jpg o .png. Puede proporcionar la imagen de entrada como una matriz de bytes de imagen (con codificación en base64) o especificar un objeto de Amazon S3. En estos procedimientos carga una imagen (.jpg o .png) en su bucket de S3.

Para ejecutar estos procedimientos, debe tener instalado el SDK AWS CLI o el correspondiente AWS . Para obtener más información, consulte Introducción a Amazon Rekognition. La cuenta de AWS que utilice debe tener permisos de acceso a la API de Amazon Rekognition. Para obtener más información, consulte Acciones definidas por Amazon Rekognition.

Para detectar etiquetas de moderación en una imagen (SDK)

1. Si aún no lo ha hecho:

   a. Cree o actualice un usuario con los permisos `AmazonRekognitionFullAccess` y `AmazonS3ReadOnlyAccess`. Para obtener más información, consulte Paso 1: Configurar una cuenta de AWS y crear un usuario.

   b. Instale y configure el AWS CLI y el AWS SDKs. Para obtener más información, consulte Paso 2: Configure y AWS CLI AWS SDKs.

2. Suba una imagen en su bucket de S3.

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

3. Consulte los siguientes ejemplos para llamar a la operación `DetectModerationLabels`.

Java

El resultado de este ejemplo son los nombres de las etiquetas de contenido inapropiado y los niveles de confianza detectados, así como la etiqueta principal de las etiquetas de moderación detectadas.

Reemplace los valores de `amzn-s3-demo-bucket` y `photo` por el nombre del bucket de S3 y el nombre de archivo de imagen que utilizó en el paso 2.

```
//Copyright 2018 Amazon.com, Inc. or its affiliates. All Rights Reserved.
//PDX-License-Identifier: MIT-0 (For details, see https://github.com/awsdocs/
amazon-rekognition-developer-guide/blob/master/LICENSE-SAMPLECODE.)

package aws.example.rekognition.image;
import com.amazonaws.services.rekognition.AmazonRekognition;
import com.amazonaws.services.rekognition.AmazonRekognitionClientBuilder;
import com.amazonaws.services.rekognition.model.AmazonRekognitionException;
import com.amazonaws.services.rekognition.model.DetectModerationLabelsRequest;
import com.amazonaws.services.rekognition.model.DetectModerationLabelsResult;
import com.amazonaws.services.rekognition.model.Image;
import com.amazonaws.services.rekognition.model.ModerationLabel;
import com.amazonaws.services.rekognition.model.S3Object;

import java.util.List;

public class DetectModerationLabels
{
   public static void main(String[] args) throws Exception
   {
      String photo = "input.jpg";
      String bucket = "bucket";

      AmazonRekognition rekognitionClient =
 AmazonRekognitionClientBuilder.defaultClient();

      DetectModerationLabelsRequest request = new
 DetectModerationLabelsRequest()
        .withImage(new Image().withS3Object(new
 S3Object().withName(photo).withBucket(bucket)))
        .withMinConfidence(60F);
      try
      {
```

```
            DetectModerationLabelsResult result =
 rekognitionClient.detectModerationLabels(request);
            List<ModerationLabel> labels = result.getModerationLabels();
            System.out.println("Detected labels for " + photo);
            for (ModerationLabel label : labels)
            {
               System.out.println("Label: " + label.getName()
                + "\n Confidence: " + label.getConfidence().toString() + "%"
                + "\n Parent:" + label.getParentName());
            }
        }
        catch (AmazonRekognitionException e)
        {
          e.printStackTrace();
        }
    }
}
```

Java V2

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

```
//snippet-start:[rekognition.java2.recognize_video_text.import]
//snippet-start:[rekognition.java2.detect_mod_labels.import]
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import software.amazon.awssdk.services.rekognition.model.Image;
import
 software.amazon.awssdk.services.rekognition.model.DetectModerationLabelsRequest;
import
 software.amazon.awssdk.services.rekognition.model.DetectModerationLabelsResponse;
import software.amazon.awssdk.services.rekognition.model.ModerationLabel;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;
import java.util.List;
//snippet-end:[rekognition.java2.detect_mod_labels.import]
```

```
/**
* Before running this Java V2 code example, set up your development environment,
 including your credentials.
*
* For more information, see the following documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
*/
public class ModerateLabels {

 public static void main(String[] args) {

     final String usage = "\n" +
         "Usage: " +
         "   <sourceImage>\n\n" +
         "Where:\n" +
         "   sourceImage - The path to the image (for example, C:\\AWS\
\pic1.png). \n\n";

     if (args.length < 1) {
         System.out.println(usage);
         System.exit(1);
     }

     String sourceImage = args[0];
     Region region = Region.US_WEST_2;
     RekognitionClient rekClient = RekognitionClient.builder()
         .region(region)
         .credentialsProvider(ProfileCredentialsProvider.create("profile-name"))
         .build();

     detectModLabels(rekClient, sourceImage);
     rekClient.close();
 }

 // snippet-start:[rekognition.java2.detect_mod_labels.main]
 public static void detectModLabels(RekognitionClient rekClient, String
 sourceImage) {

     try {
         InputStream sourceStream = new FileInputStream(sourceImage);
         SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);
```

```
        Image souImage = Image.builder()
            .bytes(sourceBytes)
            .build();

        DetectModerationLabelsRequest moderationLabelsRequest =
DetectModerationLabelsRequest.builder()
            .image(souImage)
            .minConfidence(60F)
            .build();

        DetectModerationLabelsResponse moderationLabelsResponse =
rekClient.detectModerationLabels(moderationLabelsRequest);
        List<ModerationLabel> labels =
moderationLabelsResponse.moderationLabels();
        System.out.println("Detected labels for image");

        for (ModerationLabel label : labels) {
            System.out.println("Label: " + label.name()
                + "\n Confidence: " + label.confidence().toString() + "%"
                + "\n Parent:" + label.parentName());
        }

    } catch (RekognitionException | FileNotFoundException e) {
        e.printStackTrace();
        System.exit(1);
    }
}
// snippet-end:[rekognition.java2.detect_mod_labels.main]
```

AWS CLI

Este AWS CLI comando muestra el resultado JSON de la operación `detect-moderation-labels` CLI.

Reemplace `amzn-s3-demo-bucket` y `input.jpg` por el nombre del bucket de S3 y el nombre de archivo de imagen que utilizó en el Paso 2. Sustituya el valor de `profile_name` de por el nombre de su perfil de desarrollador. Para usar un adaptador, proporcione el ARN de la versión del proyecto al parámetro `project-version`.

```
aws rekognition detect-moderation-labels --image "{S3Object:{Bucket:<amzn-s3-
demo-bucket>,Name:<image-name>}}" \
--profile profile-name \
```

```
--project-version "ARN"
```

Si accede a la CLI en un dispositivo Windows, utilice comillas dobles en lugar de comillas simples y evite las comillas dobles internas con una barra invertida (es decir, \) para corregir cualquier error del analizador que pueda encontrar. Por ver un ejemplo, consulte lo siguiente:

```
aws rekognition detect-moderation-labels --image "{\"S3Object\":{\"Bucket\":
\"amzn-s3-demo-bucket\",\"Name\":\"image-name\"}}" \
--profile profile-name
```

Python

El resultado de este ejemplo son los nombres de las etiquetas de contenido ofensivo o inapropiado y los niveles de confianza detectados, así como la etiqueta principal de las etiquetas de contenido inapropiado detectadas.

En la función `main`, reemplace los valores de `amzn-s3-demo-bucket` y `photo` por el nombre del bucket de S3 y el nombre del archivo de imagen que utilizó en el paso 2. Sustituya el valor de `profile_name` en la línea que crea la sesión de Rekognition por el nombre de su perfil de desarrollador.

```
#Copyright 2018 Amazon.com, Inc. or its affiliates. All Rights Reserved.
#PDX-License-Identifier: MIT-0 (For details, see https://github.com/awsdocs/
amazon-rekognition-developer-guide/blob/master/LICENSE-SAMPLECODE.)

import boto3

def moderate_image(photo, bucket):

    session = boto3.Session(profile_name='profile-name')
    client = session.client('rekognition')

    response = client.detect_moderation_labels(Image={'S3Object':
{'Bucket':bucket,'Name':photo}})

    print('Detected labels for ' + photo)
    for label in response['ModerationLabels']:
        print (label['Name'] + ' : ' + str(label['Confidence']))
        print (label['ParentName'])
    return len(response['ModerationLabels'])
```

```
def main():

    photo='image-name'
    bucket='amzn-s3-demo-bucket'
    label_count=moderate_image(photo, bucket)
    print("Labels detected: " + str(label_count))

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

.NET

El resultado de este ejemplo son los nombres de las etiquetas de contenido ofensivo o inapropiado y los niveles de confianza detectados, así como la etiqueta principal de las etiquetas de moderación detectadas.

Reemplace los valores de `amzn-s3-demo-bucket` y `photo` por el nombre del bucket de S3 y el nombre de archivo de imagen que utilizó en el paso 2.

```
//Copyright 2018 Amazon.com, Inc. or its affiliates. All Rights Reserved.
//PDX-License-Identifier: MIT-0 (For details, see https://github.com/awsdocs/
amazon-rekognition-developer-guide/blob/master/LICENSE-SAMPLECODE.)

using System;
using Amazon.Rekognition;
using Amazon.Rekognition.Model;

public class DetectModerationLabels
{
    public static void Example()
    {
        String photo = "input.jpg";
        String bucket = "amzn-s3-demo-bucket";

        AmazonRekognitionClient rekognitionClient = new
 AmazonRekognitionClient();

        DetectModerationLabelsRequest detectModerationLabelsRequest = new
 DetectModerationLabelsRequest()
        {
            Image = new Image()
            {
                S3Object = new S3Object()
```

```
                {
                    Name = photo,
                    Bucket = bucket
                },
            },
            MinConfidence = 60F
        };

        try
        {
            DetectModerationLabelsResponse detectModerationLabelsResponse =
 rekognitionClient.DetectModerationLabels(detectModerationLabelsRequest);
            Console.WriteLine("Detected labels for " + photo);
            foreach (ModerationLabel label in
 detectModerationLabelsResponse.ModerationLabels)
                Console.WriteLine("Label: {0}\n Confidence: {1}\n Parent: {2}",
                    label.Name, label.Confidence, label.ParentName);
        }
        catch (Exception e)
        {
            Console.WriteLine(e.Message);
        }
    }
}
```

DetectModerationLabels solicitud de operación

La entrada de `DetectModerationLabels` es una imagen. En este ejemplo de entrada de JSON, la imagen de origen se carga desde un bucket de Amazon S3. `MinConfidence` es el nivel mínimo de confianza que debe tener Amazon Rekognition Image en la precisión de la etiqueta detectada para que se devuelva en la respuesta.

```
{
    "Image": {
        "S3Object": {
            "Bucket": "amzn-s3-demo-bucket",
            "Name": "input.jpg"
        }
    },
    "MinConfidence": 60
}
```

DetectModerationLabels respuesta de operación

`DetectModerationLabels` puede recuperar imágenes de entrada desde un bucket de S3 o puede proporcionarlas como bytes de imagen. El siguiente ejemplo es la respuesta de una llamada a `DetectModerationLabels`.

En el siguiente ejemplo de respuesta de JSON, observe lo siguiente:

- Información de detección de imágenes inapropiada: el ejemplo muestra una lista de etiquetas para el contenido inapropiado u ofensivo que se encuentra en la imagen. La lista incluye la etiqueta de nivel superior y todas las etiquetas de segundo nivel que se detectan en la imagen.

  Etiqueta: cada etiqueta tiene un nombre, una estimación de la confianza que Amazon Rekognition tiene de que la etiqueta es correcta y el nombre de su etiqueta principal. El nombre de entidad principal de una etiqueta de nivel superior es "".

  Confianza de etiqueta: cada etiqueta tiene un valor de confianza comprendido entre 0 y 100 que indica el porcentaje de confianza que Amazon Rekognition tiene de que la etiqueta es correcta. El nivel de confianza necesario de una etiqueta devuelto se especifica en la respuesta de la solicitud de la operación de la API.


```
{
    "ModerationLabels": [
        {
            "Confidence": 99.44782257080078,
            "Name": "Smoking",
            "ParentName": "Drugs & Tobacco Paraphernalia & Use",
            "TaxonomyLevel": 3
        },
        {
            "Confidence": 99.44782257080078,
            "Name": "Drugs & Tobacco Paraphernalia & Use",
            "ParentName": "Drugs & Tobacco",
            "TaxonomyLevel": 2
        },
        {
            "Confidence": 99.44782257080078,
            "Name": "Drugs & Tobacco",
            "ParentName": "",
            "TaxonomyLevel": 1
        }
```

```
    ],
    "ModerationModelVersion": "7.0",
    "ContentTypes": [
        {
            "Confidence": 99.9999008178711,
            "Name": "Illustrated"
        }
    ]
}
```

Detectar vídeos almacenados inapropiados

La detección de contenido ofensivo o inapropiado en los vídeos almacenados por Amazon Rekognition Video es una operación asíncrona. Para empezar a detectar contenido inapropiado u ofensivo, llama StartContentModeration. Amazon Rekognition Video publica el estado de finalización de una operación de análisis de vídeo en un tema de Amazon Simple Notification Service. Si el análisis de vídeo es correcto, llame a GetContentModeration para obtener los resultados del análisis. Para obtener más información sobre cómo iniciar el análisis de vídeo y obtener los resultados, consulte Cómo llamar a las operaciones de Amazon Rekognition Video. Para ver una lista de las etiquetas de moderación en Amazon Rekognition, consulte Uso de la moderación de imágenes y vídeos. APIs

Este procedimiento amplía el código de Análisis de un vídeo almacenado en un bucket de Amazon S3 con Java o Python (SDK), que utiliza una cola de Amazon Simple Queue Service para obtener el estado de realización de una solicitud de análisis de vídeo.

Para detectar contenido inapropiado u ofensivo en un vídeo almacenado en un bucket de Amazon S3 (SDK)

1. Realice Análisis de un vídeo almacenado en un bucket de Amazon S3 con Java o Python (SDK).
2. Añada el código siguiente a la clase `VideoDetect` que ha creado en el paso 1.

   Java

```
    //Copyright 2018 Amazon.com, Inc. or its affiliates. All Rights Reserved.
    //PDX-License-Identifier: MIT-0 (For details, see https://github.com/
awsdocs/amazon-rekognition-developer-guide/blob/master/LICENSE-SAMPLECODE.)

        //Content moderation
 ==================================================================
```

```
       private static void StartUnsafeContentDetection(String bucket, String
video) throws Exception{

           NotificationChannel channel= new NotificationChannel()
                   .withSNSTopicArn(snsTopicArn)
                   .withRoleArn(roleArn);

           StartContentModerationRequest req = new
StartContentModerationRequest()
                   .withVideo(new Video()
                           .withS3Object(new S3Object()
                               .withBucket(bucket)
                               .withName(video)))
                   .withNotificationChannel(channel);



            StartContentModerationResult startModerationLabelDetectionResult =
rek.startContentModeration(req);
            startJobId=startModerationLabelDetectionResult.getJobId();

        }

        private static void GetUnsafeContentDetectionResults() throws
Exception{

            int maxResults=10;
            String paginationToken=null;
            GetContentModerationResult moderationLabelDetectionResult =null;

            do{
                if (moderationLabelDetectionResult !=null){
                    paginationToken =
moderationLabelDetectionResult.getNextToken();
                }

                moderationLabelDetectionResult = rek.getContentModeration(
                        new GetContentModerationRequest()
                            .withJobId(startJobId)
                            .withNextToken(paginationToken)
                            .withSortBy(ContentModerationSortBy.TIMESTAMP)
                            .withMaxResults(maxResults));
```

```
                VideoMetadata
videoMetaData=moderationLabelDetectionResult.getVideoMetadata();

                System.out.println("Format: " + videoMetaData.getFormat());
                System.out.println("Codec: " + videoMetaData.getCodec());
                System.out.println("Duration: " +
videoMetaData.getDurationMillis());
                System.out.println("FrameRate: " +
videoMetaData.getFrameRate());


                //Show moderated content labels, confidence and detection
times
                List<ContentModerationDetection> moderationLabelsInFrames=
                        moderationLabelDetectionResult.getModerationLabels();

                for (ContentModerationDetection label:
moderationLabelsInFrames) {
                    long seconds=label.getTimestamp()/1000;
                    System.out.print("Sec: " + Long.toString(seconds));
                    System.out.println(label.getModerationLabel().toString());
                    System.out.println();
                }
            } while (moderationLabelDetectionResult !=null &&
moderationLabelDetectionResult.getNextToken() != null);

        }
```

En la función `main`, reemplace las líneas:

```
        StartLabelDetection(amzn-s3-demo-bucket, video);

        if (GetSQSMessageSuccess()==true)
         GetLabelDetectionResults();
```

por:

```
        StartUnsafeContentDetection(amzn-s3-demo-bucket, video);

        if (GetSQSMessageSuccess()==true)
```

```
        GetUnsafeContentDetectionResults();
```

Java V2

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.NotificationChannel;
import software.amazon.awssdk.services.rekognition.model.S3Object;
import software.amazon.awssdk.services.rekognition.model.Video;
import
 software.amazon.awssdk.services.rekognition.model.StartContentModerationRequest;
import
 software.amazon.awssdk.services.rekognition.model.StartContentModerationResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import
 software.amazon.awssdk.services.rekognition.model.GetContentModerationResponse;
import
 software.amazon.awssdk.services.rekognition.model.GetContentModerationRequest;
import software.amazon.awssdk.services.rekognition.model.VideoMetadata;
import
 software.amazon.awssdk.services.rekognition.model.ContentModerationDetection;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class VideoDetectInappropriate {
    private static String startJobId = "";

    public static void main(String[] args) {

        final String usage = """

                Usage:    <bucket> <video> <topicArn> <roleArn>
```

```
                Where:
                   bucket - The name of the bucket in which the video is located
(for example, (for example, myBucket).\s
                   video - The name of video (for example, people.mp4).\s
                   topicArn - The ARN of the Amazon Simple Notification Service
(Amazon SNS) topic.\s
                   roleArn - The ARN of the AWS Identity and Access Management
(IAM) role to use.\s
                """;

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

       String bucket = args[0];
       String video = args[1];
       String topicArn = args[2];
       String roleArn = args[3];
       Region region = Region.US_EAST_1;
       RekognitionClient rekClient = RekognitionClient.builder()
               .region(region)
               .build();

       NotificationChannel channel = NotificationChannel.builder()
               .snsTopicArn(topicArn)
               .roleArn(roleArn)
               .build();

       startModerationDetection(rekClient, channel, bucket, video);
       getModResults(rekClient);
       System.out.println("This example is done!");
       rekClient.close();
   }

   public static void startModerationDetection(RekognitionClient rekClient,
           NotificationChannel channel,
           String bucket,
           String video) {

       try {
           S3Object s3Obj = S3Object.builder()
                   .bucket(bucket)
```

```
                    .name(video)
                    .build();

            Video vidOb = Video.builder()
                    .s3Object(s3Obj)
                    .build();

            StartContentModerationRequest modDetectionRequest =
StartContentModerationRequest.builder()
                    .jobTag("Moderation")
                    .notificationChannel(channel)
                    .video(vidOb)
                    .build();

            StartContentModerationResponse startModDetectionResult = rekClient
                    .startContentModeration(modDetectionRequest);
            startJobId = startModDetectionResult.jobId();

        } catch (RekognitionException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void getModResults(RekognitionClient rekClient) {
        try {
            String paginationToken = null;
            GetContentModerationResponse modDetectionResponse = null;
            boolean finished = false;
            String status;
            int yy = 0;

            do {
                if (modDetectionResponse != null)
                    paginationToken = modDetectionResponse.nextToken();

                GetContentModerationRequest modRequest =
GetContentModerationRequest.builder()
                        .jobId(startJobId)
                        .nextToken(paginationToken)
                        .maxResults(10)
                        .build();

                // Wait until the job succeeds.
```

```
                while (!finished) {
                    modDetectionResponse =
rekClient.getContentModeration(modRequest);
                    status = modDetectionResponse.jobStatusAsString();

                    if (status.compareTo("SUCCEEDED") == 0)
                        finished = true;
                    else {
                        System.out.println(yy + " status is: " + status);
                        Thread.sleep(1000);
                    }
                    yy++;
                }

                finished = false;

                // Proceed when the job is done - otherwise VideoMetadata is
null.
                VideoMetadata videoMetaData =
modDetectionResponse.videoMetadata();
                System.out.println("Format: " + videoMetaData.format());
                System.out.println("Codec: " + videoMetaData.codec());
                System.out.println("Duration: " +
videoMetaData.durationMillis());
                System.out.println("FrameRate: " + videoMetaData.frameRate());
                System.out.println("Job");

                List<ContentModerationDetection> mods =
modDetectionResponse.moderationLabels();
                for (ContentModerationDetection mod : mods) {
                    long seconds = mod.timestamp() / 1000;
                    System.out.print("Mod label: " + seconds + " ");
                    System.out.println(mod.moderationLabel().toString());
                    System.out.println();
                }

            } while (modDetectionResponse != null &&
modDetectionResponse.nextToken() != null);

        } catch (RekognitionException | InterruptedException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
```

```
}
```

Python

```
#Copyright 2018 Amazon.com, Inc. or its affiliates. All Rights Reserved.
#PDX-License-Identifier: MIT-0 (For details, see https://github.com/awsdocs/
amazon-rekognition-developer-guide/blob/master/LICENSE-SAMPLECODE.)

    # ============== Unsafe content ===============
    def StartUnsafeContent(self):
        response=self.rek.start_content_moderation(Video={'S3Object': {'Bucket':
 self.bucket, 'Name': self.video}},
            NotificationChannel={'RoleArn': self.roleArn, 'SNSTopicArn':
 self.snsTopicArn})

        self.startJobId=response['JobId']
        print('Start Job Id: ' + self.startJobId)

    def GetUnsafeContentResults(self):
        maxResults = 10
        paginationToken = ''
        finished = False

        while finished == False:
            response = self.rek.get_content_moderation(JobId=self.startJobId,
                                                MaxResults=maxResults,
                                                NextToken=paginationToken,
                                                SortBy="NAME",
                                                AggregateBy="TIMESTAMPS")

            print('Codec: ' + response['VideoMetadata']['Codec'])
            print('Duration: ' + str(response['VideoMetadata']
['DurationMillis']))
            print('Format: ' + response['VideoMetadata']['Format'])
            print('Frame rate: ' + str(response['VideoMetadata']['FrameRate']))
            print()

            for contentModerationDetection in response['ModerationLabels']:
                print('Label: ' +
                    str(contentModerationDetection['ModerationLabel']['Name']))
                print('Confidence: ' +
                    str(contentModerationDetection['ModerationLabel']
['Confidence']))
```

```
                print('Parent category: ' +
                    str(contentModerationDetection['ModerationLabel']
['ParentName']))
                print('Timestamp: ' +
 str(contentModerationDetection['Timestamp']))
                print()

            if 'NextToken' in response:
                paginationToken = response['NextToken']
            else:
                finished = True
```

En la función `main`, reemplace las líneas:

```
    analyzer.StartLabelDetection()
    if analyzer.GetSQSMessageSuccess()==True:
        analyzer.GetLabelDetectionResults()
```

por:

```
    analyzer.StartUnsafeContent()
    if analyzer.GetSQSMessageSuccess()==True:
        analyzer.GetUnsafeContentResults()
```

> **Note**
>
> Si ya ha ejecutado un ejemplo de vídeo distinto de Análisis de un vídeo almacenado en un bucket de Amazon S3 con Java o Python (SDK), el código que se va a reemplazar podría ser diferente.

3. Ejecute el código. Se muestra una lista de etiquetas de contenido inapropiado detectadas en el vídeo.

GetContentModeration respuesta de operación

La respuesta de `GetContentModeration` es una matriz, `ModerationLabels` de objetos ContentModerationDetection. La matriz contiene un elemento por cada vez que se detecta una etiqueta de contenido inapropiado. Dentro de un `ContentModerationDetectionObject` objeto,

ModerationLabelcontiene información sobre un elemento detectado de contenido inapropiado u ofensivo. Timestampes el tiempo, en milisegundos desde el inicio del vídeo, en el que se detectó la etiqueta. Las etiquetas se organizan jerárquicamente de la misma forma que las etiquetas detectadas durante los análisis de la imagen de contenido inapropiado. Para obtener más información, consulte Moderación del contenido.

El siguiente es un ejemplo de respuesta de GetContentModeration, ordenada por NAME y agregada por TIMESTAMPS.

```
{
    "JobStatus": "SUCCEEDED",
    "VideoMetadata": {
        "Codec": "h264",
        "DurationMillis": 54100,
        "Format": "QuickTime / MOV",
        "FrameRate": 30.0,
        "FrameHeight": 462,
        "FrameWidth": 884,
        "ColorRange": "LIMITED"
    },
    "ModerationLabels": [
        {
            "Timestamp": 36000,
            "ModerationLabel": {
                "Confidence": 52.451576232910156,
                "Name": "Alcohol",
                "ParentName": "",
                "TaxonomyLevel": 1
            },
            "ContentTypes": [
                {
                    "Confidence": 99.9999008178711,
                    "Name": "Animated"
                }
            ]
        },
        {
            "Timestamp": 36000,
            "ModerationLabel": {
                "Confidence": 52.451576232910156,
                "Name": "Alcoholic Beverages",
                "ParentName": "Alcohol",
                "TaxonomyLevel": 2
```

```
            },
            "ContentTypes": [
                {
                    "Confidence": 99.9999008178711,
                    "Name": "Animated"
                }
            ]
        }
    ],
    "ModerationModelVersion": "7.0",
    "JobId": "a1b2c3d4...",
    "Video": {
        "S3Object": {
            "Bucket": "amzn-s3-demo-bucket",
            "Name": "video-name.mp4"
        }
    },
    "GetRequestMetadata": {
        "SortBy": "TIMESTAMP",
        "AggregateBy": "TIMESTAMPS"
    }
}
```

El siguiente es un ejemplo de respuesta de `GetContentModeration`, ordenada por `NAME` y agregada por `SEGMENTS`.

```
{
    "JobStatus": "SUCCEEDED",
    "VideoMetadata": {
        "Codec": "h264",
        "DurationMillis": 54100,
        "Format": "QuickTime / MOV",
        "FrameRate": 30.0,
        "FrameHeight": 462,
        "FrameWidth": 884,
        "ColorRange": "LIMITED"
    },
    "ModerationLabels": [
        {
            "Timestamp": 0,
            "ModerationLabel": {
                "Confidence": 0.0003000000142492354,
                "Name": "Alcohol Use",
```

```
                "ParentName": "Alcohol",
                "TaxonomyLevel": 2
            },
            "StartTimestampMillis": 0,
            "EndTimestampMillis": 29520,
            "DurationMillis": 29520,
            "ContentTypes": [
                {
                    "Confidence": 99.9999008178711,
                    "Name": "Illustrated"
                },
                {
                    "Confidence": 99.9999008178711,
                    "Name": "Animated"
                }
            ]
        }
    ],
    "ModerationModelVersion": "7.0",
    "JobId": "a1b2c3d4...",
    "Video": {
        "S3Object": {
            "Bucket": "amzn-s3-demo-bucket",
            "Name": "video-name.mp4"
        }
    },
    "GetRequestMetadata": {
        "SortBy": "TIMESTAMP",
        "AggregateBy": "SEGMENTS"
    }
}
```

Mejora de la precisión con la moderación personalizada

La DetectModerationLabelsAPI de Amazon Rekognition le permite detectar contenido inapropiado, no deseado u ofensivo. La función de moderación personalizada de Rekognition le permite mejorar la precisión mediante el uso de adaptadores. DetectModerationLabels Los adaptadores son componentes modulares que se pueden añadir a un modelo de aprendizaje profundo de Rekognition existente, lo que amplía sus capacidades para las tareas en las que está entrenado. Al crear un adaptador y proporcionárselo a la DetectModerationLabelsoperación, puede lograr una mayor precisión en las tareas de moderación de contenido relacionadas con su caso de uso específico.

Al personalizar el modelo de moderación de contenido de Rekognition para etiquetas de moderación específicas, debe crear un proyecto y entrenar un adaptador a partir del conjunto de imágenes que usted proporcione. A continuación, puede comprobar el rendimiento del adaptador de forma iterativa y volver a entrenarlo hasta el nivel de precisión deseado. Los proyectos se utilizan para contener las diferentes versiones de los adaptadores.

Puede utilizar la consola de Rekognition para crear proyectos y adaptadores. Como alternativa, puede utilizar un AWS SDK y el correspondiente APIs para crear un proyecto, entrenar un adaptador y administrar sus adaptadores.

Crear y usar adaptadores

Los adaptadores son componentes modulares que se pueden añadir al modelo de aprendizaje profundo de Rekognition existente, lo que amplía sus capacidades para las tareas en las que está entrenado. Al entrenar un modelo de aprendizaje profundo con adaptadores, puede lograr una mayor precisión en las tareas de análisis de imágenes relacionadas con su caso de uso específico.

Para crear y usar un adaptador, debe proporcionar datos de entrenamiento y pruebas a Rekognition. Puede lograr esto usando uno de estos métodos:

- Análisis y verificación masivos: puede crear un conjunto de datos de entrenamiento analizando en masa las imágenes que Rekognition analizará y a las que asignará etiquetas. A continuación, puede revisar las anotaciones generadas para sus imágenes y verificar o corregir las predicciones. Para obtener más información sobre cómo funciona el análisis masivo de imágenes, consulte Análisis masivo.
- Anotación manual: con este enfoque, crea sus datos de entrenamiento cargando y anotando imágenes. Los datos de prueba se crean cargando y anotando imágenes o dividiéndolas automáticamente.

Elija uno de los siguientes temas para obtener más información:

Temas

- Análisis y verificación masivos
- Anotación manual

Análisis y verificación masivos

Con este enfoque, sube una gran cantidad de imágenes que desea usar como datos de entrenamiento y luego usa Rekognition para obtener predicciones para estas imágenes, que les asigna etiquetas automáticamente. Puede utilizar estas predicciones como punto de partida para el adaptador. Puede verificar la precisión de las predicciones y, a continuación, entrenar el adaptador en función de las predicciones verificadas. Esto se puede hacer con la AWS consola.

Análisis masivo y Moderación personalizada

Subida de imágenes para su análisis masivo

Para crear un conjunto de datos de entrenamiento para su adaptador, suba imágenes de forma masiva para que Rekognition pueda predecir las etiquetas. Para obtener los mejores resultados, proporcione tantas imágenes para el entrenamiento como sea posible, hasta el límite de 10 000, y asegúrese de que las imágenes sean representativas de todos los aspectos de su caso de uso.

Al utilizar la AWS consola, puede cargar imágenes directamente desde su ordenador o proporcionar un depósito de Amazon Simple Storage Service que almacene sus imágenes. Sin embargo, cuando utilice APIs Rekognition con un SDK, debe proporcionar un archivo de manifiesto que haga referencia a las imágenes almacenadas en un depósito de Amazon Simple Storage Service. Para obtener más información, consulte Análisis masivo.

Revisar predicciones

Una vez que haya cargado sus imágenes a la consola de Rekognition, generará etiquetas para ellas. A continuación, puede verificar las predicciones en una de las siguientes categorías: positivo verdadero, positivo falso, negativo verdadero, negativo falso. Una vez que haya verificado las predicciones, puede entrenar un adaptador a partir de sus comentarios.

Entrenamiento del adaptador

Una vez que haya terminado de verificar las predicciones obtenidas mediante el análisis masivo, podrá iniciar el proceso de entrenamiento del adaptador.

Obtenga el AdapterId

Una vez que se haya entrenado el adaptador, podrá obtener el identificador único para usarlo en el análisis de imágenes de Rekognition. APIs

Llamar a la operación de la API

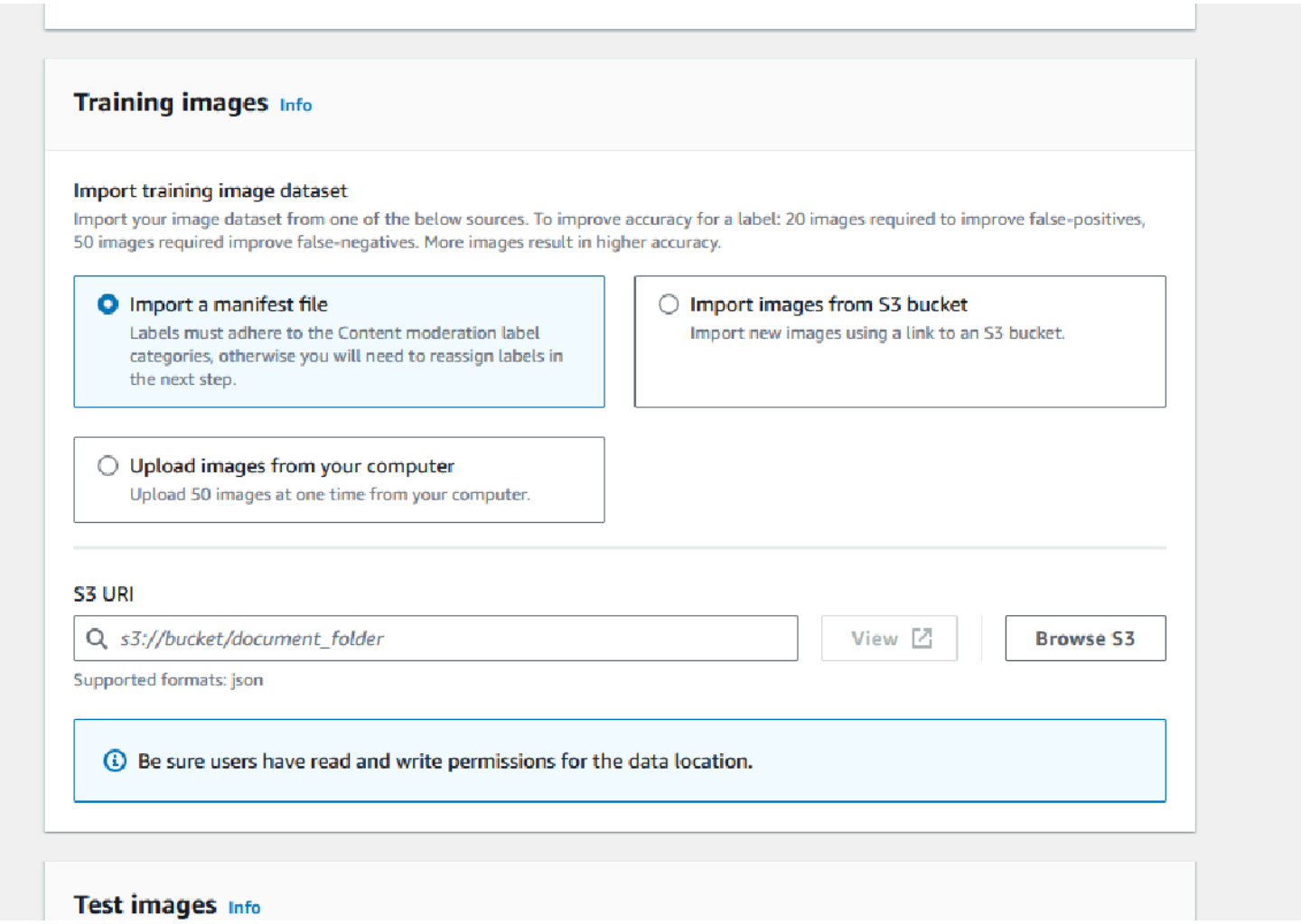
Para aplicar su adaptador personalizado, proporcione su ID cuando llame a uno de los APIs analizadores de imágenes compatibles con los adaptadores. Esto mejora la precisión de las predicciones de las imágenes.

Anotación manual

Con este enfoque, puede crear sus datos de entrenamiento cargando y anotando las imágenes manualmente. Los datos de las pruebas se crean cargando y anotando las imágenes de las pruebas o dividiéndolas automáticamente para que Rekognition utilice automáticamente una parte de los datos de entrenamiento como imágenes de prueba.

Carga y anotación de imágenes

Para entrenar el adaptador, tendrá que subir un conjunto de imágenes de muestra representativas de su caso de uso. Para obtener los mejores resultados, proporcione tantas imágenes para el entrenamiento como sea posible, hasta el límite de 10 000, y asegúrese de que las imágenes sean representativas de todos los aspectos de su caso de uso.



Al utilizar la AWS consola, puede cargar imágenes directamente desde su ordenador, proporcionar un archivo de manifiesto o proporcionar un bucket de Amazon S3 que almacene sus imágenes.

Sin embargo, cuando utilice APIs Rekognition con un SDK, debe proporcionar un archivo de manifiesto que haga referencia a las imágenes almacenadas en un bucket de Amazon S3.

Puede usar la interfaz de anotación de la consola de Rekognition para anotar sus imágenes. Anote sus imágenes etiquetándolas con etiquetas, así establecerá una «verdad básica» para el entrenamiento. También debe designar conjuntos de entrenamiento y prueba, o usar la característica de división automática, antes de poder entrenar un adaptador. Cuando termine de designar los conjuntos de datos y anotar las imágenes, puede crear un adaptador basado en las imágenes anotadas del conjunto de pruebas. A continuación, puede evaluar el rendimiento del adaptador.

Creación de conjunto de pruebas

Deberá proporcionar un conjunto de pruebas anotado o utilizar la característica de división automática. El conjunto de entrenamiento se utiliza para entrenar el adaptador. El adaptador aprende los patrones contenidos en estas imágenes anotadas. El conjunto de prueba se utiliza para evaluar el rendimiento del modelo antes de finalizar el adaptador.

Entrenamiento del adaptador

Una vez que haya terminado de anotar los datos de entrenamiento o haya proporcionado un archivo de manifiesto, puede iniciar el proceso de entrenamiento del adaptador.

Obtención del ID del adaptador

Una vez que se haya entrenado el adaptador, podrá obtener el identificador único del adaptador para usarlo en el análisis de imágenes de Rekognition. APIs

Llamar a la operación de la API

Para aplicar su adaptador personalizado, proporcione su ID cuando llame a uno de los APIs analizadores de imágenes compatibles con los adaptadores. Esto mejora la precisión de las predicciones de las imágenes.

Preparación de los conjuntos de datos de entrada

La creación de un adaptador requiere que proporcione a Rekognition dos conjuntos de datos, un conjunto de datos de entrenamiento y un conjunto de datos de prueba. Cada conjunto de datos se compone de dos elementos: imágenes y anotaciones/etiquetas. En las siguientes secciones se explica para qué se utilizan las etiquetas y las imágenes y cómo se combinan para crear conjuntos de datos.

Imágenes

Necesitará entrenar un adaptador con muestras representativas de sus imágenes. Cuando seleccione imágenes para el entrenamiento, intente incluir al menos algunas imágenes que demuestren la respuesta esperada para cada una de las etiquetas a las que se dirige con el adaptador.

Para crear un conjunto de datos de entrenamiento, debe proporcionar uno de los dos tipos de imágenes siguientes:

- Imágenes con predicciones de falsos positivos. Por ejemplo, cuando un modelo base predice que una imagen contiene alcohol, pero no es así.
- Imágenes con predicciones de falsos negativos. Por ejemplo, cuando un modelo base predice que una imagen no contiene alcohol, pero sí lo contiene.

Para crear un conjunto de datos equilibrado, se recomienda proporcionar uno de los dos tipos de imágenes siguientes:

- Imágenes con predicciones verdaderamente positivas. Por ejemplo, cuando un modelo base predice correctamente que una imagen contiene alcohol. Se recomienda proporcionar estas imágenes si proporciona imágenes de falsos positivos.
- Imágenes con predicciones de verdaderos negativos. Por ejemplo, cuando un modelo base predice correctamente que una imagen no contiene alcohol. Se recomienda proporcionar estas imágenes si proporciona imágenes de falsos negativos.

Etiquetas

Una etiqueta hace referencia a cualquiera de los siguientes elementos: objetos, eventos, conceptos o actividades. En el caso de la moderación de contenido, una etiqueta es una instancia de contenido inapropiado, no deseado u ofensivo.

En el contexto de la creación de un adaptador mediante el entrenamiento del modelo base de Rekognition, cuando se asigna una etiqueta a una imagen, se denomina anotación. Cuando entrene un adaptador con la consola de Rekognition, utilizará la consola para añadir anotaciones a las imágenes; para ello, deberá elegir una etiqueta y, a continuación, etiquetar las imágenes que se correspondan con la etiqueta. Mediante este proceso, el modelo aprende a identificar los elementos de las imágenes en función de la etiqueta asignada. Este proceso de vinculación permite que el modelo se centre en el contenido más relevante cuando se crea un adaptador, lo que mejora la precisión del análisis de imágenes.

Como alternativa, puede proporcionar un archivo de manifiesto que contenga información sobre las imágenes y las anotaciones que las acompañan.

Conjuntos de datos de entrenamiento y prueba

El conjunto de datos de entrenamiento es la base para ajustar el modelo y crear un adaptador personalizado. Debe proporcionar un conjunto de datos de entrenamiento anotado para que el

modelo pueda aprender. El modelo aprende de este conjunto de datos para mejorar su rendimiento en el tipo de imágenes que usted proporciona.

Para mejorar la precisión, debes crear tu conjunto de datos de entrenamiento a partir de annotation/labeling imágenes. Puede lograr esto de dos maneras:

- Asignación manual de etiquetas: puede usar la consola Rekognition para crear un conjunto de datos de entrenamiento cargando las imágenes que quiere que contenga su conjunto de datos y, a continuación, asignarles etiquetas manualmente.
- Archivo de manifiesto: puede utilizar un archivo de manifiesto para entrenar el adaptador. El archivo de manifiesto contiene información sobre las anotaciones más precisas para sus imágenes de entrenamiento y pruebas, así como la ubicación de las imágenes de entrenamiento. Puede proporcionar el archivo de manifiesto cuando entrene un adaptador con APIs Rekognition o cuando utilice la consola. AWS

El conjunto de datos de prueba se utiliza para evaluar el rendimiento del adaptador después del entrenamiento. Para garantizar una evaluación fiable, el conjunto de datos de prueba se crea utilizando una parte del conjunto de datos de entrenamiento original que el modelo no haya visto antes. Este proceso garantiza que el rendimiento del adaptador se evalúe con nuevos datos, lo que crea mediciones y métricas precisas. Para obtener mejoras de precisión óptimas, consulte Prácticas recomendadas de adaptadores de entrenamiento.

Administración de adaptadores con la AWS CLI y SDKs

Rekognition le permite utilizar múltiples características que aprovechan los modelos de visión artificial previamente entrenados. Con estos modelos, puede llevar a cabo tareas como la detección de etiquetas y la moderación del contenido. También puede personalizar estos modelos específicos con un adaptador.

Puede utilizar la creación y gestión de proyectos de Rekognition APIs (como CreateProjecty CreateProjectVersion) para crear y entrenar adaptadores. En las siguientes páginas se describe cómo usar las operaciones de la API para crear, entrenar y administrar los adaptadores mediante la AWS consola, el AWS SDK que elija o la AWS CLI.

Después de entrenar un adaptador, puede usarlo para realizar inferencias con las características compatibles. Actualmente, los adaptadores son compatibles cuando se utiliza la característica de moderación de contenido.

Cuando entrenes un adaptador con un AWS SDK, debes proporcionar las etiquetas de información básica (anotaciones de imagen) en forma de archivo de manifiesto. También puede utilizar la consola de Rekognition para crear y entrenar un adaptador.

> **Note**
>
> Los adaptadores no se pueden copiar. Solo se pueden copiar las versiones de los proyectos de Etiquetas personalizadas de Rekognition.

Temas

- Estados de adaptador
- Creación de un proyecto
- Descripción de proyectos
- Eliminación de un proyecto
- Creación de una versión del proyecto
- Describir una versión del proyecto
- Eliminación de una versión del proyecto

Estados de adaptador

El adaptador de Moderación personalizada (versiones de los proyectos) puede tener uno de los siguientes estados:

- TRAINING_IN_PROGRESS: el adaptador está en proceso de entrenamiento con los archivos que ha proporcionado como documentos de entrenamiento.
- TRAINING_COMPLETED: el adaptador ha completado correctamente el entrenamiento y está listo para que revise su rendimiento.
- TRAINING_FAILED: el adaptador no ha podido completar su entrenamiento por algún motivo. Revise el archivo del manifiesto de salida y el resumen del manifiesto de salida para obtener información sobre la causa del error.
- DELETING: el adaptador está en proceso de eliminarse.
- DEPRECATED: el adaptador se ha entrenado con una versión anterior del modelo base de Moderación de contenido. Está en periodo de gracia y expirará entre 60 y 90 días a

partir del lanzamiento de la nueva versión del modelo base. Durante el período de gracia, puedes seguir utilizando el adaptador para realizar inferencias DetectModerationLabelso StartMediaAnalysisJobrealizar operaciones de API. Consulte la consola de Moderación personalizada para ver la fecha de vencimiento de los adaptadores.

- CADUCADO: el adaptador se entrenó con una versión anterior del modelo base de moderación de contenido y ya no se puede utilizar para obtener resultados personalizados con las operaciones de la StartMediaAnalysisJob API DetectModerationLabels o de la API. Si se especifica un adaptador con el estado Expired en una solicitud de inferencia, este se ignorará y, en su lugar, se devolverá la respuesta de la versión más reciente del modelo base de Moderación personalizada.

Creación de un proyecto

Con esta CreateProjectoperación, puede crear un proyecto que contenga un adaptador para las operaciones de detección de etiquetas de Rekognition. Un proyecto es un grupo de recursos y, en el caso de operaciones de detección de etiquetas, por ejemplo DetectModerationLabels, un proyecto te permite almacenar adaptadores que puedes usar para personalizar el modelo base de Rekognition. Al invocar CreateProject, se proporciona al argumento el nombre del proyecto que se quiere crear. ProjectName

Para crear un proyecto con la AWS consola:

- Inicie sesión en la consola de Rekognition
- Haga clic en Moderación personalizada
- Elija Crear proyecto
- Seleccione Crear un nuevo proyecto o Agregar a un proyecto existente
- Agregue un nombre de proyecto
- Agregue un nombre de adaptador
- Agregue una descripción si lo desea
- Elija cómo quiere importar las imágenes de su entrenamiento: archivo de manifiesto, desde un bucket de S3 o desde su ordenador
- Elija si quiere dividir automáticamente sus datos de entrenamiento o importar un archivo de manifiesto
- Seleccione si quiere que el proyecto se actualice automáticamente o no
- Haga clic en Crear proyecto

Para crear un proyecto con la AWS CLI y el SDK:

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

CLI

```
# Request
# Creating Content Moderation Project
aws rekognition create-project \
    --project-name "project-name" \
    --feature CONTENT_MODERATION \
    --auto-update ENABLED
    --profile profile-name
```

Descripción de proyectos

Puede usar la DescribeProjectsAPI para obtener información sobre sus proyectos, incluida la información sobre todos los adaptadores asociados a un proyecto.

Para describir los proyectos con la AWS CLI y el SDK:

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

CLI

```
# Request
# Getting CONTENT_MODERATION project details
aws rekognition describe-projects \
    --features CONTENT_MODERATION
    --profile profile-name
```

Eliminación de un proyecto

Puedes eliminar un proyecto mediante la consola de Rekognition o llamando a la API. DeleteProject Para eliminar un proyecto, primero debe eliminar todos los adaptadores asociados. No se puede recuperar un proyecto o modelo después de eliminarlo.

Para eliminar un proyecto con la consola: AWS

- Inicie sesión en la consola de Rekognition.
- Haga clic en Moderación personalizada.
- Debe eliminar cada adaptador asociado al proyecto para poder eliminar el proyecto en sí. Elimine todos los adaptadores asociados al proyecto seleccionando el adaptador y, a continuación, seleccionando Eliminar.
- Seleccione el proyecto y, a continuación, pulse el botón Eliminar.

Para eliminar un proyecto con la AWS CLI y el SDK:

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

CLI

```
aws rekognition delete-project
  --project-arn project_arn \
  --profile profile-name
```

Creación de una versión del proyecto

Puede entrenar un adaptador para su implementación mediante la CreateProjectVersionoperación. CreateProjectVersion primero crea una nueva versión de un adaptador asociado a un proyecto y, a continuación, comienza a entrenarlo. La respuesta de CreateProjectVersion es un nombre de recurso de Amazon (ARN) para la versión del modelo. El entrenamiento tarda un tiempo en completarse. Puede obtener el estado actual llamando DescribeProjectVersions. Al entrenar un modelo, Rekognition usa los conjuntos de datos de entrenamiento y prueba asociados al proyecto.

Para crear conjuntos de datos, deberá utilizar la consola. Para obtener más información, consulte la sección sobre conjuntos de datos.

Para crear un proyecto con la consola de Rekognition:

- Inicie sesión en la consola de AWS Rekognition
- Haga clic en Moderación personalizada
- Seleccionar un proyecto.
- En la página Detalles del proyecto, elija Crear adaptador
- En la página Crear un proyecto, rellene los detalles necesarios para Detalles del proyecto, Imágenes de entrenamiento e Imágenes de prueba y, a continuación, seleccione Crear proyecto.
- En la página Asignar etiquetas a las imágenes, añada etiquetas a sus imágenes y, cuando termine, seleccione Comenzar a entrenar

Para crear una versión del proyecto con la AWS CLI y el SDK:

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

CLI

```
# Request
aws rekognition create-project-version \
 --project-arn project-arn \
 --training-data '{Assets=[GroundTruthManifest={S3Object="amzn-s3-demo-source-
bucket",Name="manifest.json"}]}' \
 --output-config S3Bucket=amzn-s3-demo-destination-bucket,S3KeyPrefix=my-results \
 --feature-config "ContentModeration={ConfidenceThreshold=70}"
 --profile profile-name
```

Describir una versión del proyecto

Puede enumerar y describir los adaptadores asociados a un proyecto mediante la DescribeProjectVersionsoperación. Puede especificar hasta 10 versiones de modelo en

ProjectVersionArns. Si no indica un valor, se devolverán las descripciones de todas las versiones del modelo en el proyecto.

Para describir la versión de un proyecto con la AWS CLI y el SDK:

1. Si aún no lo ha hecho, instale y configure la AWS CLI y el AWS SDKs. Para obtener más información, consulte Paso 2: Configure y AWS CLI AWS SDKs .
2. Utilice el siguiente código para describir una versión del proyecto:

CLI

```
aws rekognition describe-project-versions
  --project-arn project_arn \
  --version-names [versions]
```

Eliminación de una versión del proyecto

Puede eliminar un adaptador de Rekognition asociado a un proyecto mediante esta operación. DeleteProjectVersion No puede eliminar un adaptador si se está ejecutando o si se está entrenando. Para comprobar el estado de un adaptador, llame a la DescribeProjectVersions operación y compruebe el campo de estado que devuelve. Para detener una llamada al adaptador en ejecución StopProjectVersion. Si el modelo se está entrenando, espere a que termine de entrenarse para eliminarlo. Debe eliminar cada adaptador asociado al proyecto para poder eliminar el proyecto en sí.

Para eliminar un proyecto con la consola de Rekognition:

- Inicie sesión en la consola de Rekognition
- Haga clic en Moderación personalizada
- En la pestaña Proyectos, puede ver todos sus proyectos y los adaptadores asociados. Seleccione un adaptador y, a continuación, seleccione Eliminar.

Para eliminar una versión del proyecto con la AWS CLI y el SDK:

1. Si aún no lo ha hecho, instale y configure la AWS CLI y el AWS SDKs. Para obtener más información, consulte Paso 2: Configure y AWS CLI AWS SDKs .

2. Utilice el código siguiente para eliminar una versión del proyecto:

CLI

```
# Request
aws rekognition delete-project-version
  --project-version-arn model_arn \
  --profile profile-name
```

Tutorial sobre el adaptador de moderación personalizado

Este tutorial le muestra cómo crear, entrenar, evaluar, usar y administrar adaptadores mediante la consola de Rekognition. Para crear, usar y administrar adaptadores con el AWS SDK, consulteAdministración de adaptadores con la AWS CLI y SDKs.

Los adaptadores le permiten mejorar la precisión de las operaciones de la API de Rekognition y personalizar el comportamiento del modelo para adaptarlo a sus propias necesidades y casos de uso. Después de crear un adaptador con este tutorial, podrás usarlo para analizar tus propias imágenes con operaciones como DetectModerationLabels, por ejemplo, volver a entrenar el adaptador para futuras mejoras.

En este tutorial, aprenderá a:

- Crear un proyecto con la consola de Rekognition
- Anotar sus datos de entrenamiento
- Entrenar su adaptador en su conjunto de datos de entrenamiento
- Revisar el rendimiento de su adaptador
- Utilizar su adaptador para el análisis de imágenes

Requisitos previos

Antes de completar este tutorial, se recomienda que lo lea detenidamente Crear y usar adaptadores.

Para crear un adaptador, puede usar la consola de Rekognition para crear un proyecto, subir y anotar sus propias imágenes y, a continuación, entrenar un adaptador con estas imágenes. Para empezar, consulte Creación de un proyecto y entrenamiento de un adaptador.

Como alternativa, puede utilizar la consola o la API de Rekognition para recuperar las predicciones de las imágenes y, a continuación, verificarlas antes de entrenar un adaptador para utilizarlas. Para empezar, consulte Análisis masivo, verificación de predicciones y entrenamiento de un adaptador.

Anotación de imágenes

Puede anotar las imágenes usted mismo etiquetándolas con la consola de Rekognition o utilizar el análisis masivo de Rekognition para anotar las imágenes y comprobar que se han etiquetado correctamente. Elija uno de los siguientes temas para empezar.

Temas

- Creación de un proyecto y entrenamiento de un adaptador
- Análisis masivo, verificación de predicciones y entrenamiento de un adaptador

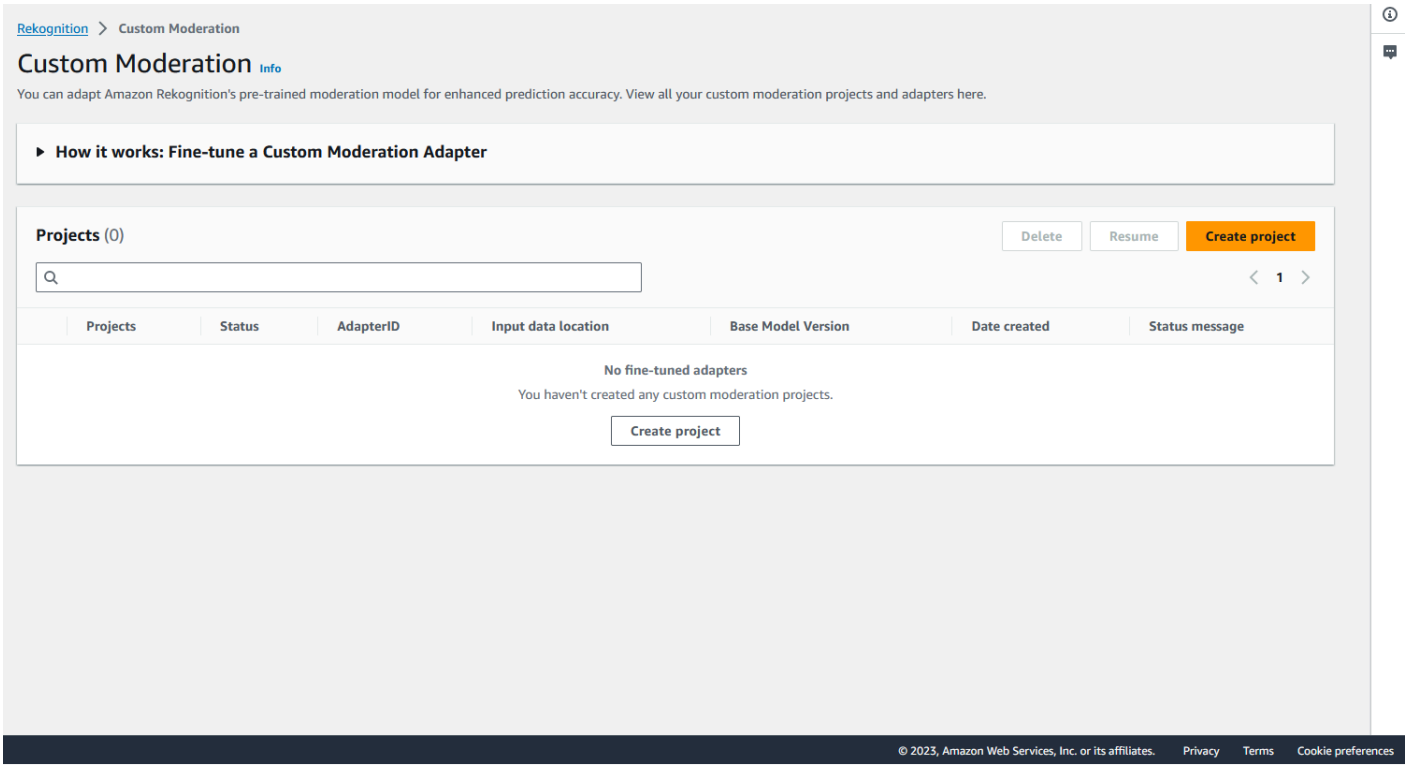
Creación de un proyecto y entrenamiento de un adaptador

Siga los pasos que se indican a continuación para entrenar el adaptador mediante la anotación de imágenes mediante la consola de Rekognition.

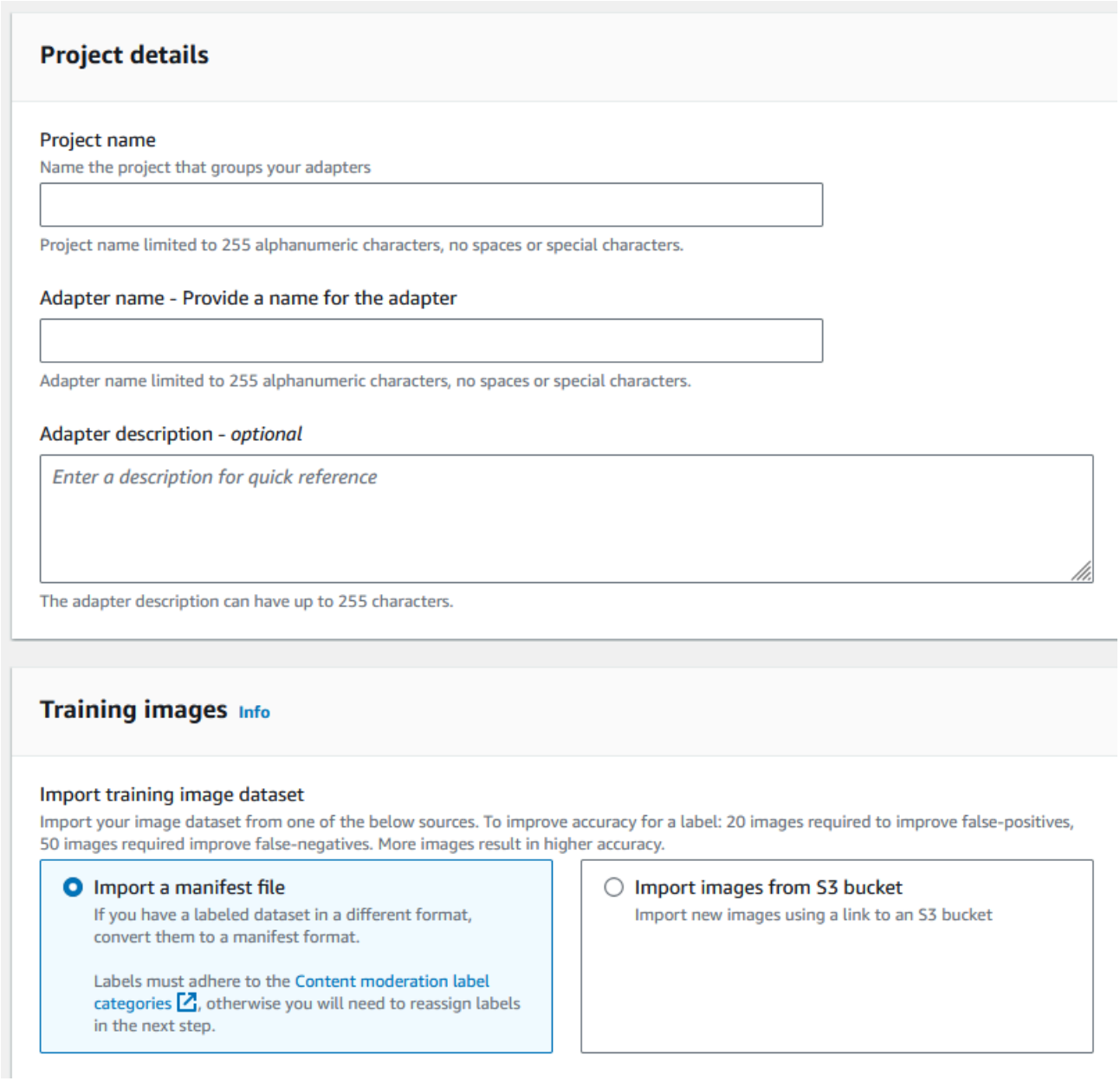
Crear un proyecto

Antes de poder entrenar o usar un adaptador, debe crear el proyecto que lo contendrá. También debe proporcionar las imágenes utilizadas para entrenar el adaptador. Para crear un proyecto, un adaptador y sus conjuntos de datos de imágenes:

1. Inicie sesión en la consola AWS de administración y abra la consola de Rekognition en. https://console.aws.amazon.com/rekognition/
2. En el panel izquierdo, elija Moderación personalizada. Aparece la página de inicio de Moderación personalizada de Rekognition.



3. La página de inicio de Moderación personalizada le muestra una lista de todos sus proyectos y adaptadores, y también hay un botón para crear un adaptador. Elija Crear proyecto para crear un nuevo proyecto y un adaptador.
4. Si es la primera vez que crea un adaptador, se le pedirá que cree un bucket de Amazon S3 para almacenar los archivos relacionados con su proyecto y su adaptador. Elija Crear bucket de Amazon S3.
5. En la página siguiente, introduzca el nombre del adaptador y el nombre del proyecto. Proporcione una descripción del adaptador si lo desea.



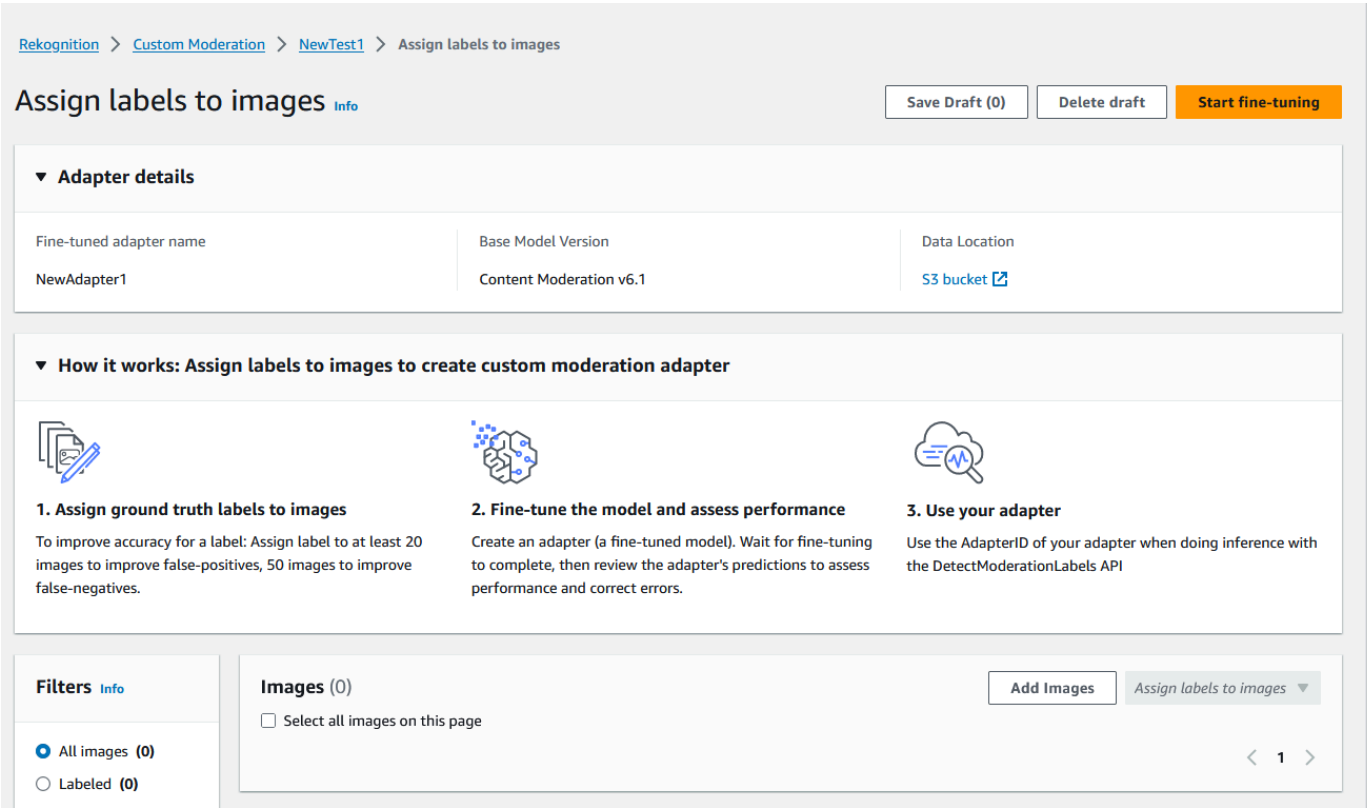
6. En este paso, también proporcionará las imágenes para el adaptador. Puede seleccionar: Importar imágenes desde su ordenador, Importar un archivo de manifiesto o Importar imágenes desde un bucket de Amazon S3. Si decide importar sus imágenes desde un bucket de Amazon S3, indica la ruta al bucket y la carpeta que contienen las imágenes de entrenamiento. Si sube las imágenes directamente desde su ordenador, tenga en cuenta que solo puede subir hasta 30 imágenes a la vez. Si utiliza un archivo de manifiesto que contiene anotaciones, puede saltarse los pasos que se indican a continuación relacionados con la anotación de imágenes y pasar a la sección sobre Revisión del rendimiento del adaptador.

7. En la sección Detalles del conjunto de datos de prueba, elija División automática para que Rekognition seleccione automáticamente el porcentaje adecuado de sus imágenes como datos de prueba, o puede elegir Importar manualmente el archivo de manifiesto.
8. Tras rellenar esta información, selecciona Crear proyecto.

Entrena un adaptador

Para entrenar un adaptador a partir de sus propias imágenes sin anotaciones:

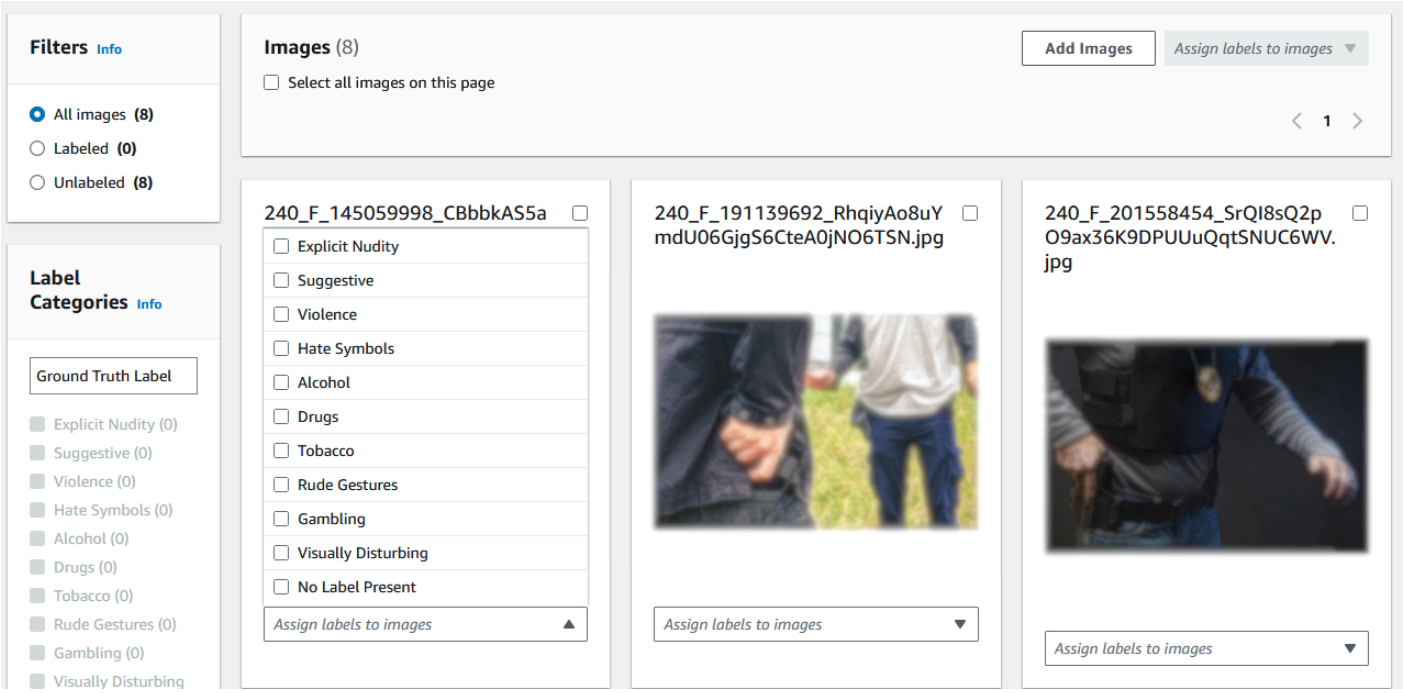
1. Seleccione el proyecto que contiene el adaptador y, a continuación, elija la opción Asignar etiqueta a las imágenes.
2. En la página Asignar etiqueta a las imágenes, puede ver todas las imágenes que se han cargado como imágenes de entrenamiento. Puede filtrar estas imágenes tanto por labeled/unlabeled estado como por categoría de etiqueta mediante los dos paneles de selección de atributos de la izquierda. Puedes añadir imágenes adicionales a su conjunto de datos de entrenamiento seleccionando el botón Añadir imágenes.



3. Tras añadir imágenes al conjunto de datos de entrenamiento, debe anotarlas con etiquetas. Tras subir las imágenes, la página «Asignar etiquetas a las imágenes» se actualizará para mostrar las imágenes que haya subido. Se le pedirá que seleccione la etiqueta adecuada para sus imágenes

de una lista desplegable de etiquetas compatibles con la moderación de Rekognition. Puede seleccionar más de una etiqueta.

4. Continúe con este proceso hasta que haya agregado etiquetas a cada una de las imágenes de sus datos de entrenamiento.
5. Después de etiquetar todos los datos, seleccione Comenzar a entrenar para empezar a entrenar el modelo, con lo que se crea el adaptador.



6. Antes de iniciar el proceso de entrenamiento, puede añadir las etiquetas que desee al adaptador. También puede proporcionar al adaptador una clave de cifrado personalizada o utilizar una clave AWS KMS. Cuando haya terminado de añadir las etiquetas que desee y de personalizar el cifrado a su gusto, seleccione Entrenar adaptador para iniciar el proceso de entrenamiento de su adaptador.
7. Espere a que el adaptador termine de entrenarse. Una vez finalizado el entrenamiento, recibirá una notificación de que el adaptador ha terminado de crearse.

Cuando el estado del adaptador sea “Entrenamiento finalizado”, podrá revisar las métricas del adaptador

Análisis masivo, verificación de predicciones y entrenamiento de un adaptador

Complete los siguientes pasos para entrenar su adaptador verificando las predicciones de los análisis masivos a partir del modelo de moderación de contenido de Rekognition.

Para entrenar un adaptador mediante la verificación de las predicciones del modelo de moderación de contenido de Rekognition, debe:

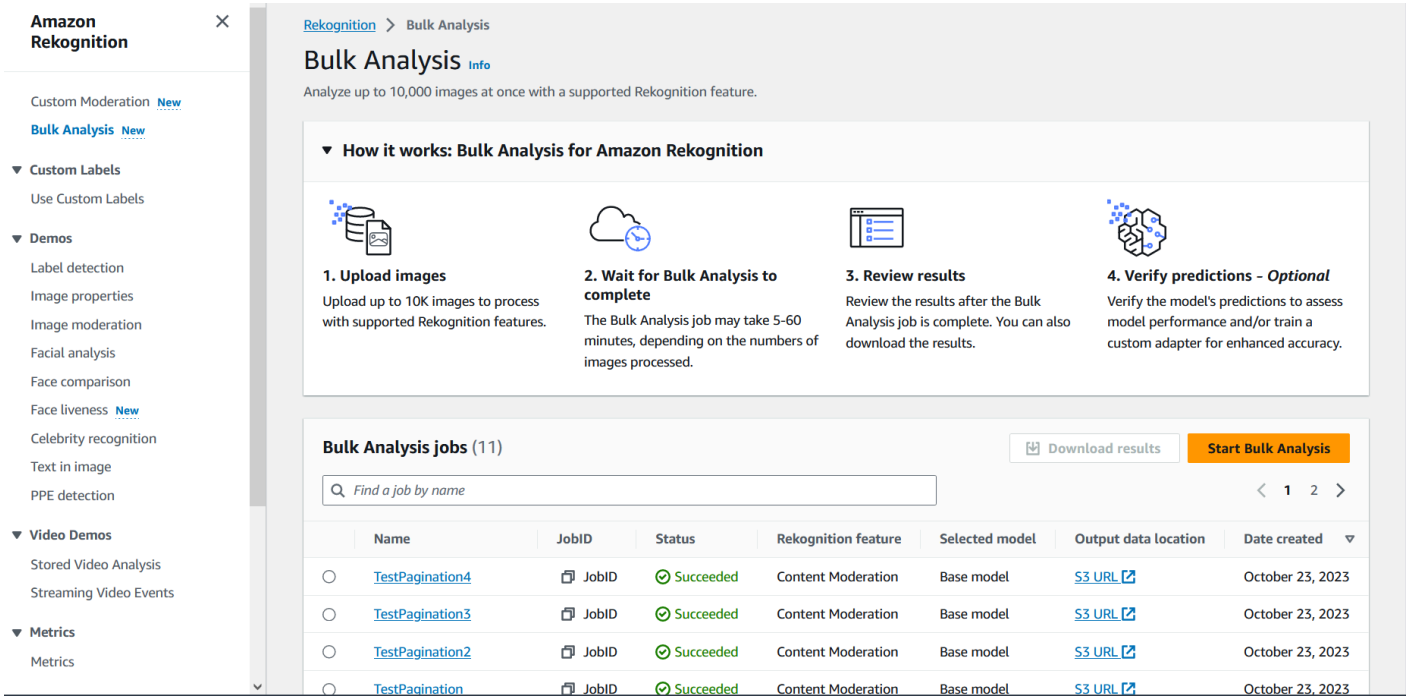
1. Realizar un análisis masivo de sus imágenes
2. Verificar las predicciones devueltas para sus imágenes

Puede obtener predicciones para las imágenes realizando un análisis masivo con el modelo base de Rekognition o con un adaptador que ya haya creado.

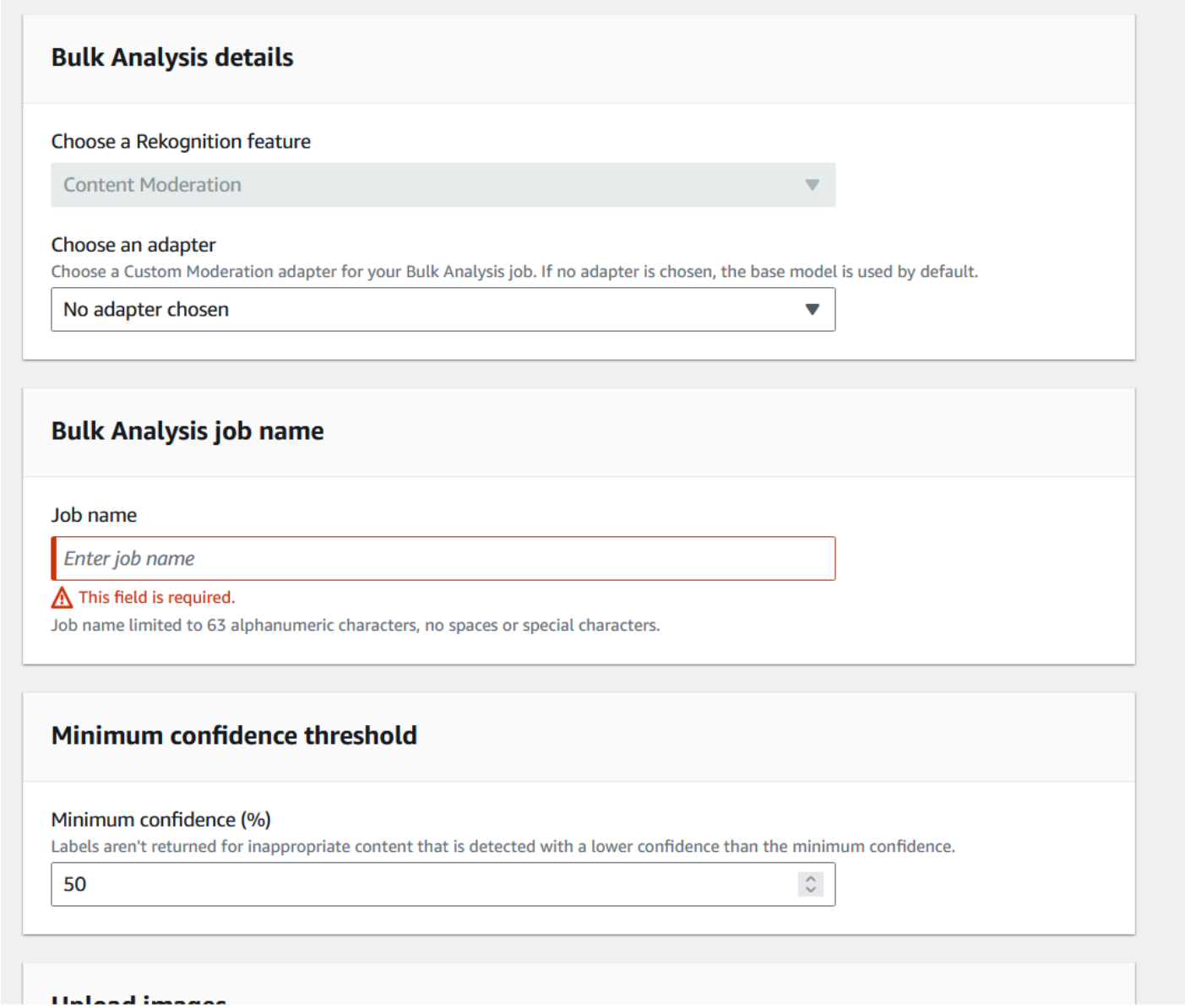
Realizar un análisis masivo de sus imágenes

Para entrenar un adaptador con las predicciones que ha verificado, primero debe iniciar un trabajo de análisis masivo para analizar un lote de imágenes utilizando el modelo base de Rekognition o un adaptador de su elección. Para ejecutar un trabajo de análisis masivo:

1. Inicie sesión en la consola Amazon Rekognition Consola de administración de AWS y ábrala en. https://console.aws.amazon.com/rekognition/
2. En el panel izquierdo, elija Análisis masivo. Aparece la página de inicio del análisis masivo. Seleccione Iniciar un análisis masivo. Se muestra información general de la característica Análisis masivo con pasos para cargar imágenes, esperar el análisis, revisar los resultados y, opcionalmente, verificar las predicciones del modelo. Muestra los trabajos recientes de Análisis masivo para Moderación de contenido con el modelo base.



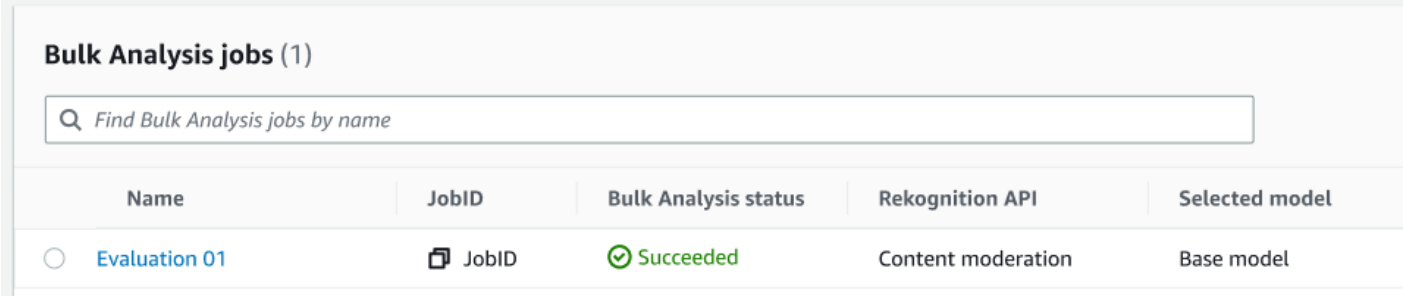
3. Si es la primera vez que crea un adaptador, se le pedirá que cree un bucket de Amazon Simple Storage Service para almacenar los archivos relacionados con su proyecto y su adaptador. Elija Crear bucket de Amazon S3.
4. Seleccione el adaptador que desee usar para el análisis masivo mediante el menú desplegable Elegir un adaptador. Si no se selecciona ningún adaptador, se utilizará el modelo base por defecto. Para este tutorial, no seleccione un adaptador.



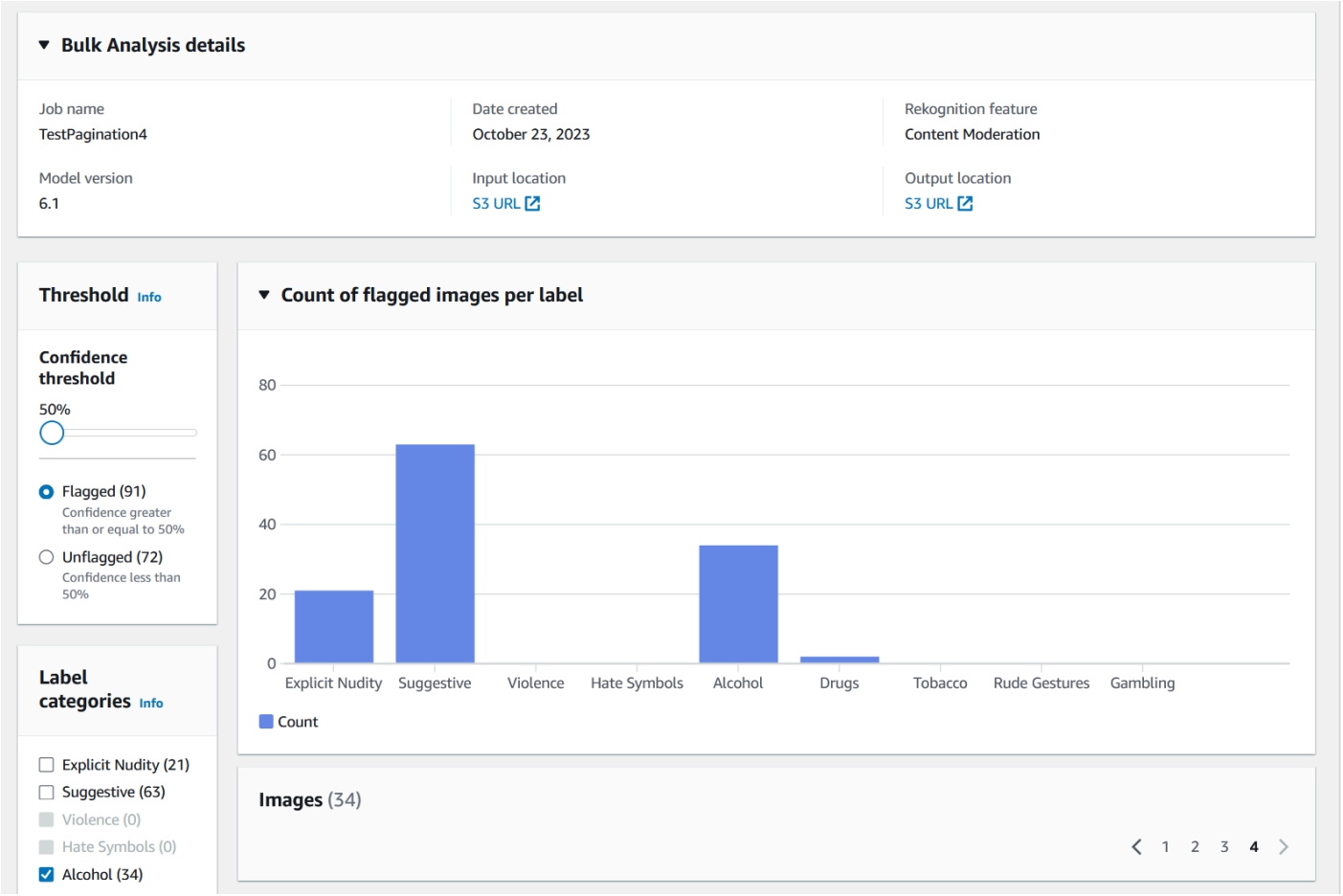
5. En el campo Nombre del trabajo de análisis masivo, rellene el nombre del trabajo de análisis masivo.
6. Elija un valor para el Umbral de confianza mínimo. No se devolverán las predicciones de etiquetas que sean inferiores al umbral de confianza elegido. Tenga en cuenta que cuando evalúe el

rendimiento del modelo más adelante, no podrá ajustar el umbral de confianza por debajo del umbral de confianza mínimo que haya elegido.

7. En este paso, también proporcionará las imágenes que desee analizar con el análisis masivo. Estas imágenes también se pueden usar para entrenar el adaptador. Puede elegir Cargar imágenes desde su ordenador o Importar imágenes desde un bucket de Amazon S3. Si decide importar sus documentos desde un bucket de Amazon S3, indica la ruta al bucket y la carpeta que contienen las imágenes de entrenamiento. Si sube los documentos directamente desde su ordenador, tenga en cuenta que solo puede subir hasta 50 imágenes a la vez.
8. Tras rellenar esta información, seleccione Iniciar análisis. Esto iniciará el proceso de análisis utilizando el modelo base de Rekognition.
9. Puede comprobar el estado de su trabajo de análisis masivo consultando el estado del análisis masivo del trabajo en la página principal de análisis masivo. Cuando el estado del análisis masivo pase a ser “Correcto”, los resultados del análisis estarán listos para su revisión.



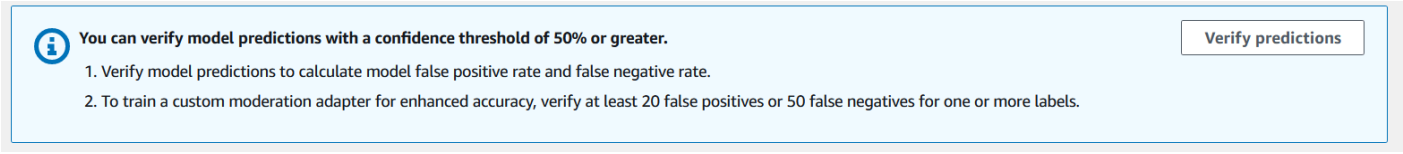
10. Elija el análisis que creó de la lista de trabajos de análisis masivo.
11. En la página de detalles del análisis masivo, puede ver las predicciones que el modelo base de Rekognition ha realizado para las imágenes que ha subido.
12. Revise el rendimiento del modelo base. Puede cambiar el umbral de confianza que debe tener el adaptador para asignar una etiqueta a una imagen mediante el control deslizante del umbral de confianza. El número de instancias marcadas y no marcadas cambiará a medida que ajuste el umbral de confianza. El panel Categorías de etiquetas muestra las categorías de nivel superior que Rekognition reconoce y puede seleccionar una categoría de esta lista para mostrar cualquier imagen a la que se le haya asignado esa etiqueta.



Compruebe las predicciones

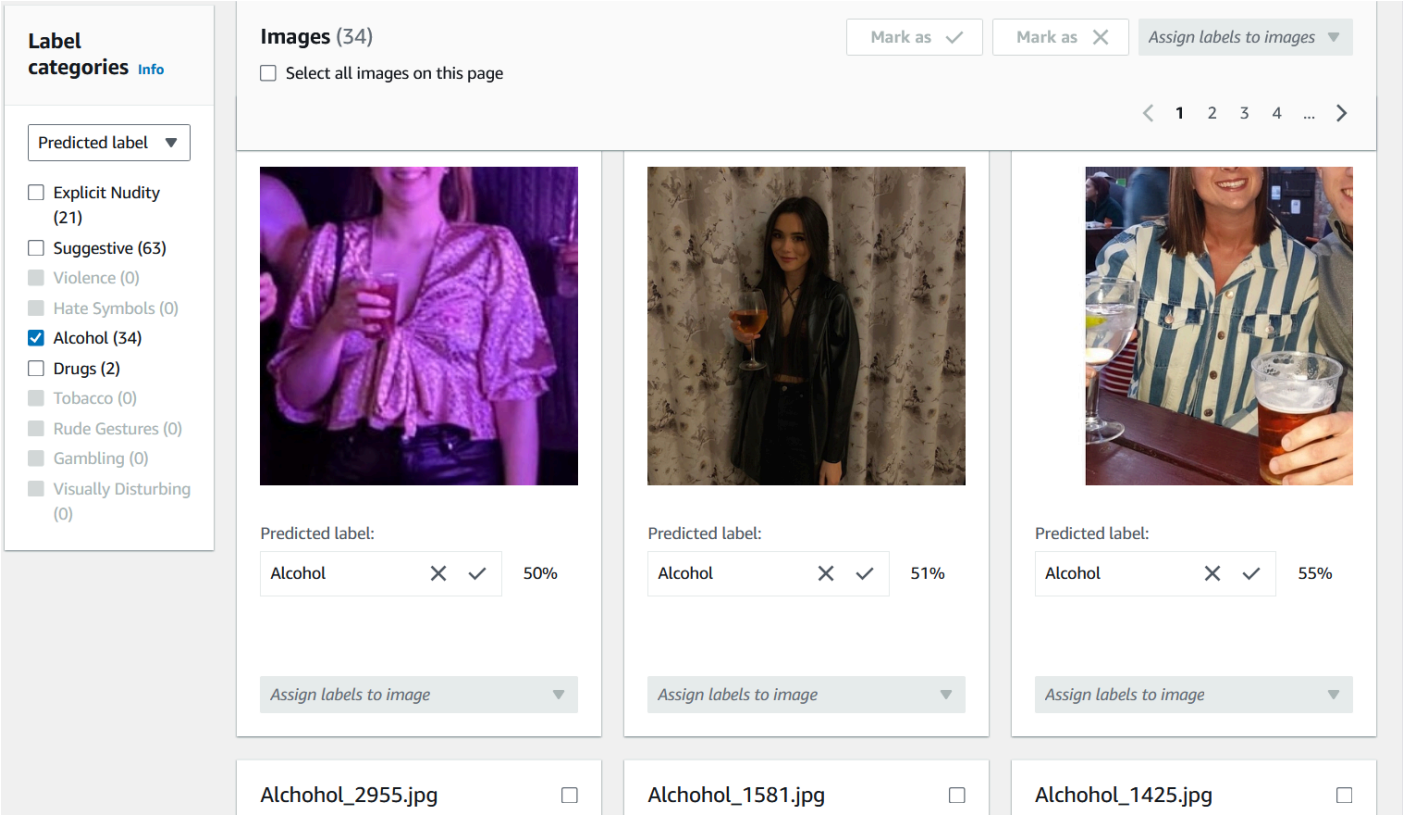
Si ha revisado la precisión del modelo base de Rekognition o de un adaptador elegido y desea mejorarla, puede utilizar el flujo de trabajo de verificación:

1. Cuando haya terminado de revisar el rendimiento del modelo base, querrá verificar las predicciones. Si corrige las predicciones, podrá entrenar un adaptador. Seleccione Verificar las predicciones en la parte superior de la página de análisis masivo.

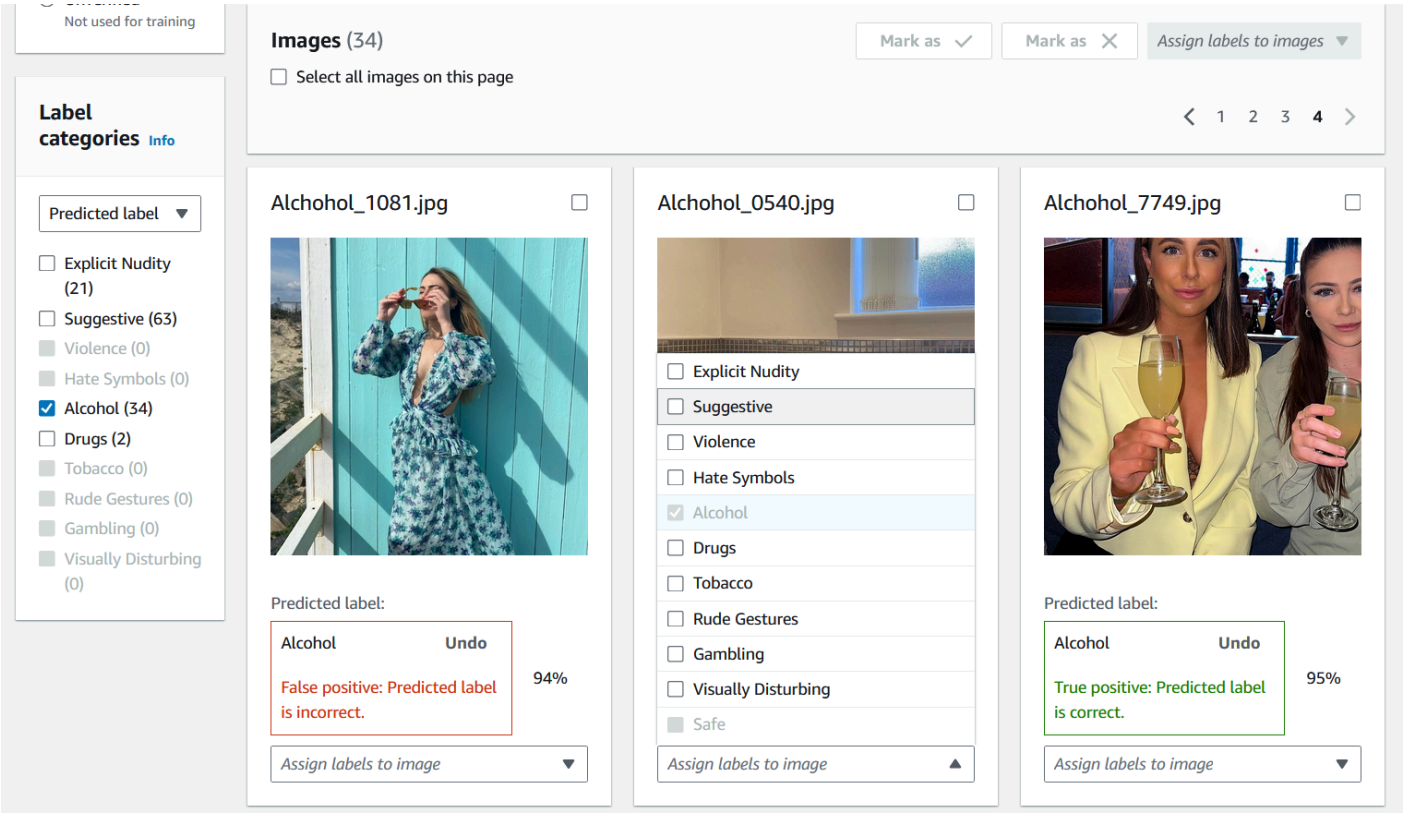


2. En la página Verificar las predicciones, puede ver todas las imágenes que proporcionó para el modelo base de Rekognition, o el adaptador elegido, junto con la etiqueta de predicción para cada imagen. Debe comprobar si cada predicción es correcta o incorrecta mediante los botones situados debajo de la imagen. Utilice el botón “X” para marcar una predicción como incorrecta y

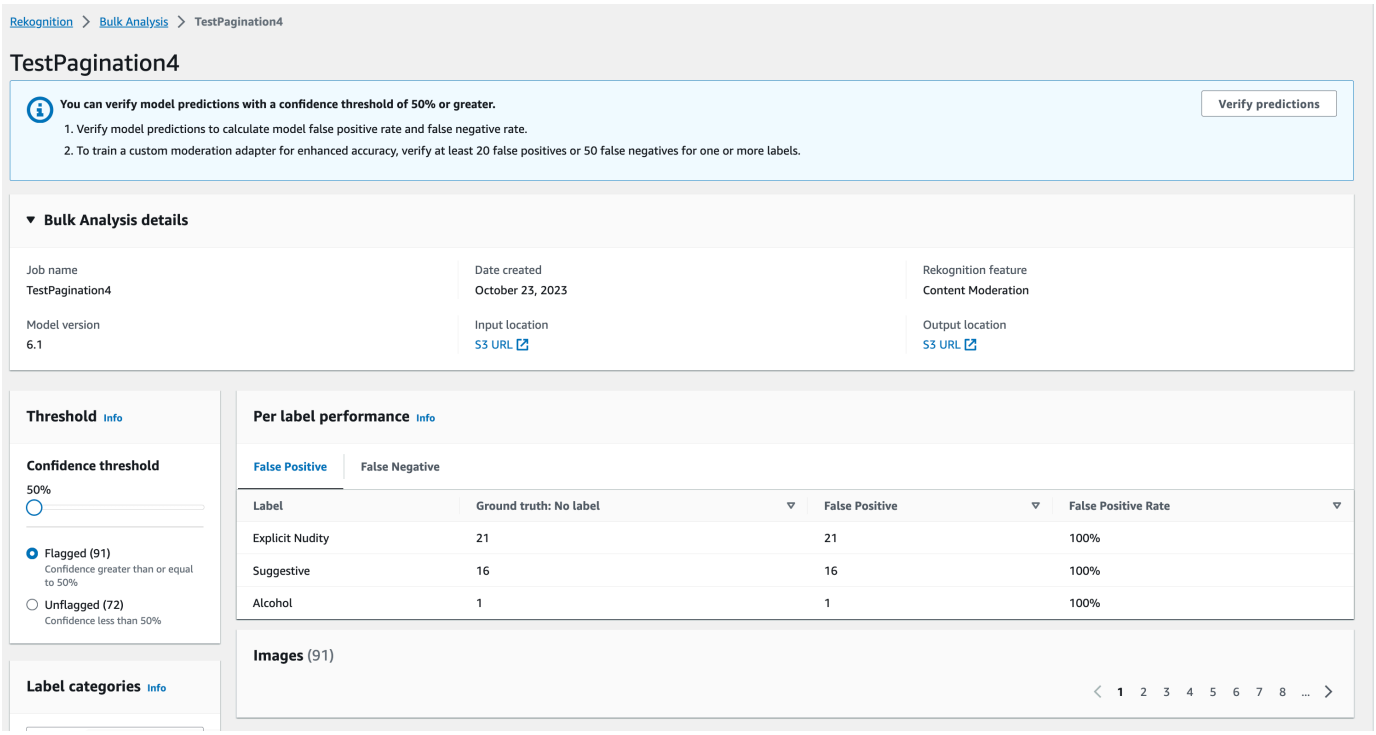
el botón de verificación para marcar una predicción como correcta. Para entrenar un adaptador, necesitará verificar al menos 20 predicciones de falsos positivos y 50 predicciones de falsos negativos para una etiqueta determinada. Cuantas más predicciones verifique, mejor será el rendimiento del adaptador.



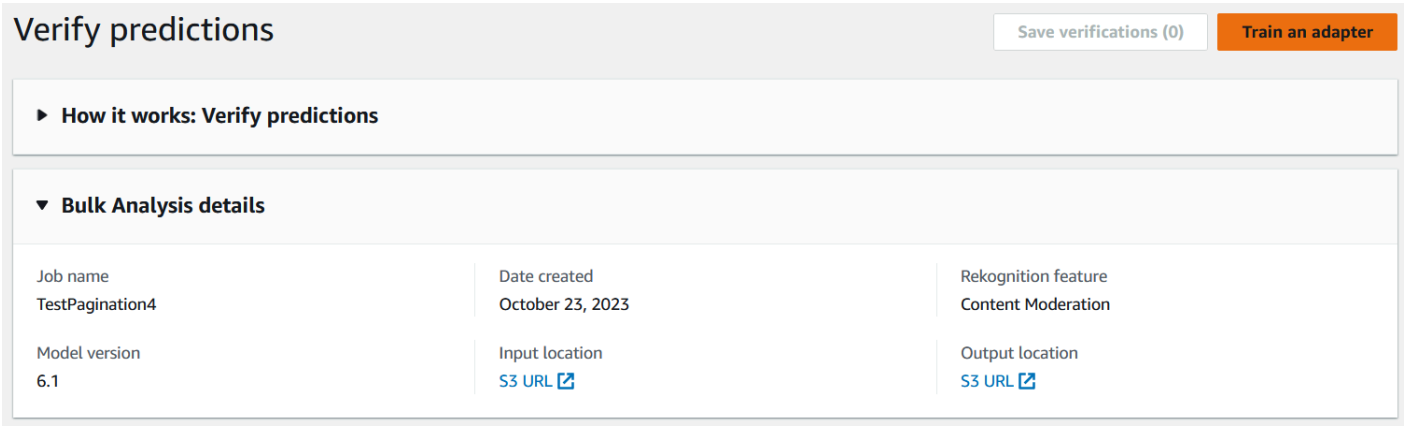
Después de verificar una predicción, el texto que aparece debajo de la imagen cambiará para mostrarte el tipo de predicción que ha verificado. Una vez que haya verificado una imagen, también puede añadir etiquetas adicionales a la imagen mediante el menú Asignar etiquetas a la imagen. Puede ver qué imágenes están marcadas o no marcadas por el modelo según el umbral de confianza que haya elegido o filtrar las imágenes por categoría.



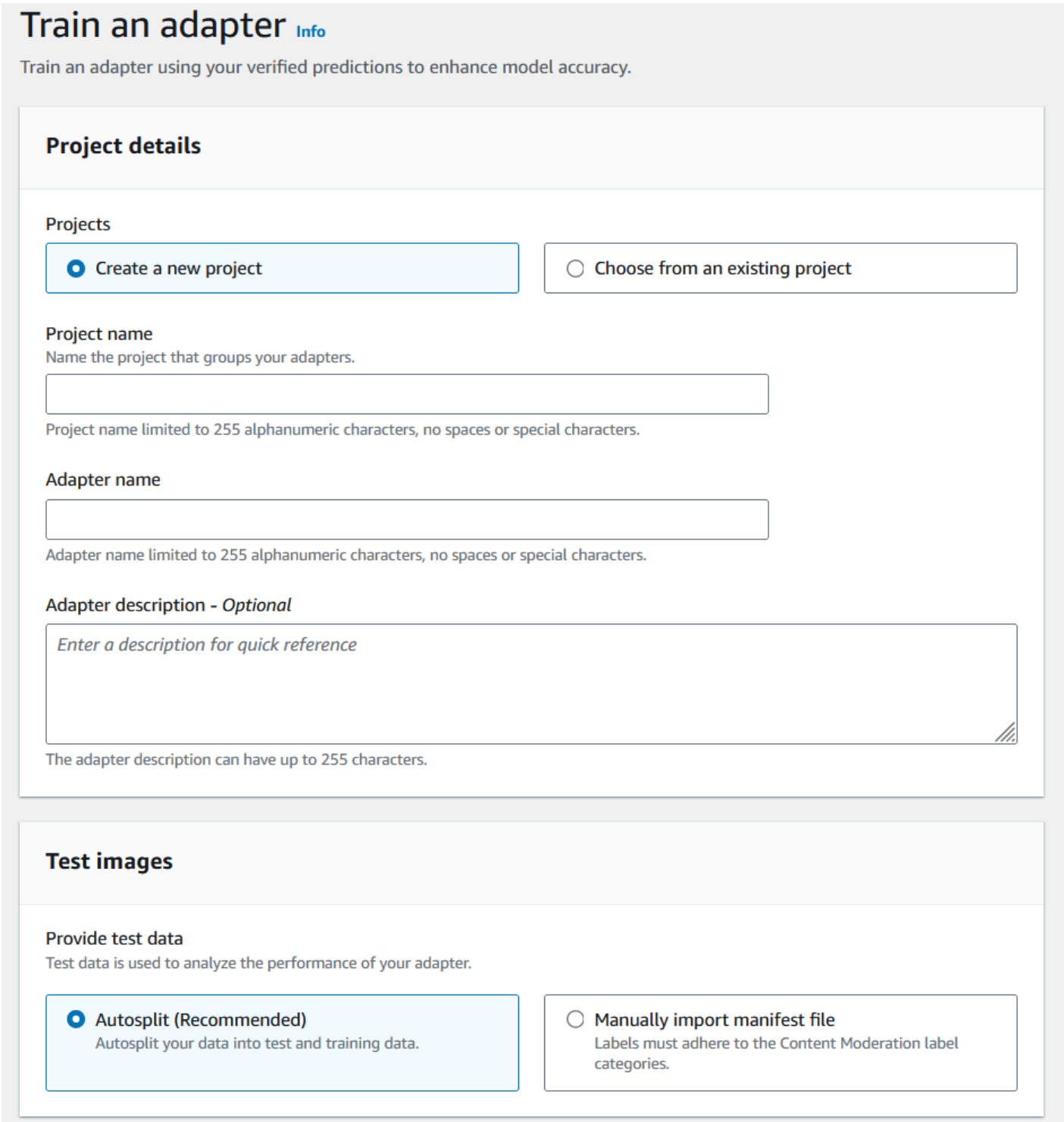
3. Cuando haya terminado de verificar todas las predicciones que desee verificar, podrá ver las estadísticas sobre las predicciones verificadas en la sección Rendimiento por etiqueta de la página de verificación. También puede volver a la página de detalles del análisis masivo para ver estas estadísticas.



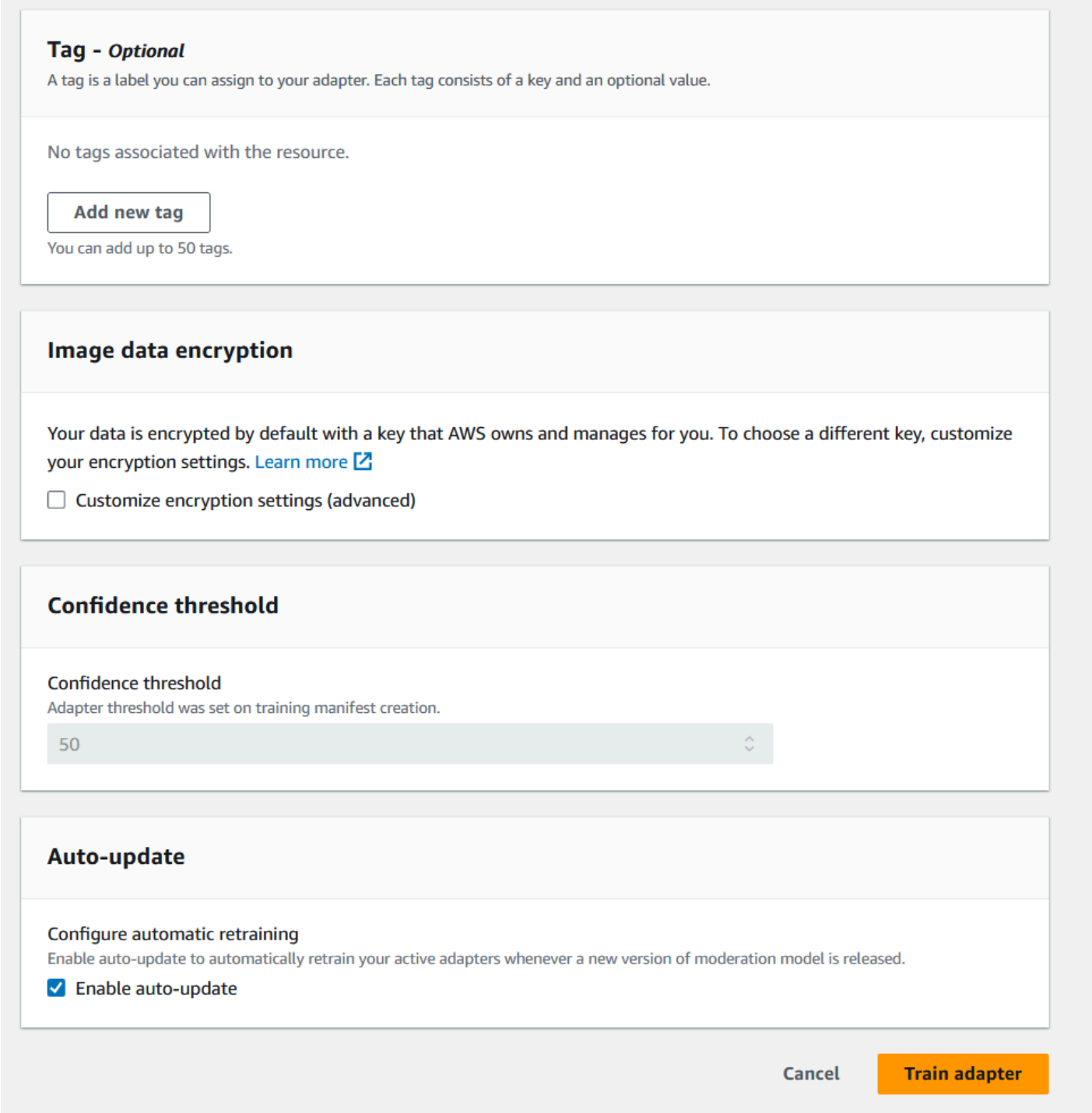
4. Cuando el resultado de las estadísticas sobre el rendimiento por etiqueta sea satisfactorio, vuelva a ir a la página Verificación de predicciones y, a continuación, pulse el botón Entrenar un adaptador para empezar a entrenar el adaptador.



5. En la página Entrenar un adaptador, se le pedirá que cree un proyecto o que elija uno existente. Asigne un nombre al proyecto y al adaptador que incluirá el proyecto. También debe especificar el origen de las imágenes de prueba. Al especificar las imágenes, puede elegir División automática para que Rekognition utilice automáticamente una parte de sus datos de entrenamiento como imágenes de prueba, o puede especificar manualmente un archivo de manifiesto. Se recomienda elegir División automática.



6. Especifique las etiquetas que desee, así como una AWS KMS clave si no desea utilizar la clave predeterminada. AWS Se recomienda dejar activada la actualización automática.
7. Elija Entrenar adaptador.

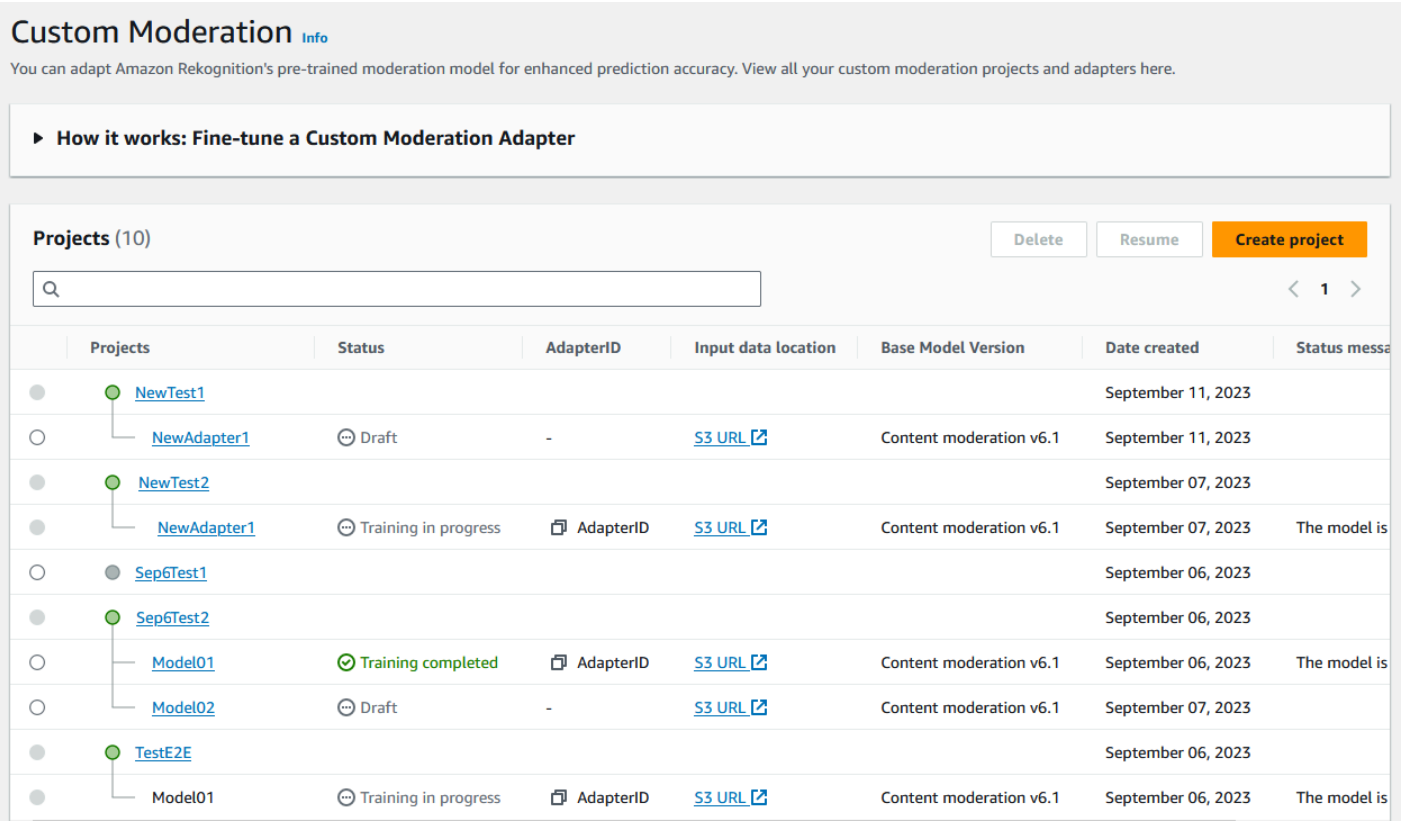


8. Cuando el estado del adaptador en la página de inicio de Moderación personalizada pase a ser «Entrenamiento ha finalizado», podrá revisar el rendimiento del adaptador. Para obtener más información, consulte Revisión del rendimiento del adaptador.

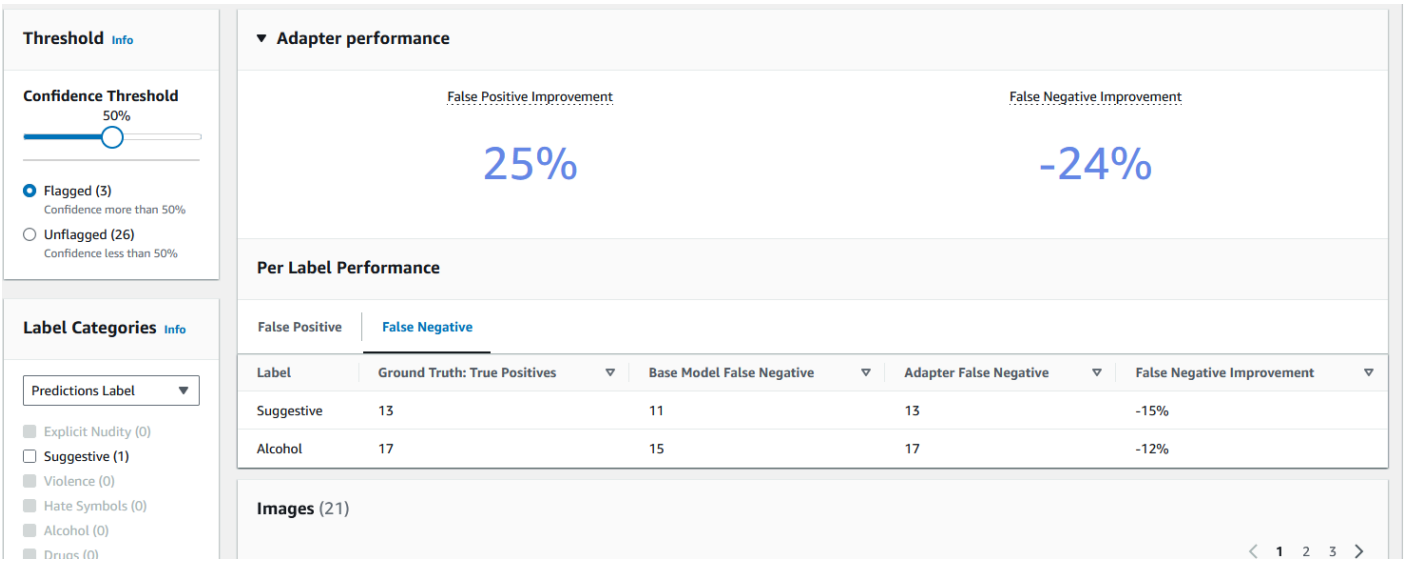
Revisión del rendimiento del adaptador

Para revisar el rendimiento del adaptador:

1. Cuando utilice la consola, podrá ver el estado de todos los adaptadores asociados a un proyecto en la pestaña Proyectos de la página de inicio de Moderación personalizada. Vaya a la página de inicio de Moderación personalizada.



2. Seleccione de esta lista el adaptador que desee revisar. En la siguiente página de detalles del adaptador, puede ver una variedad de métricas del adaptador.

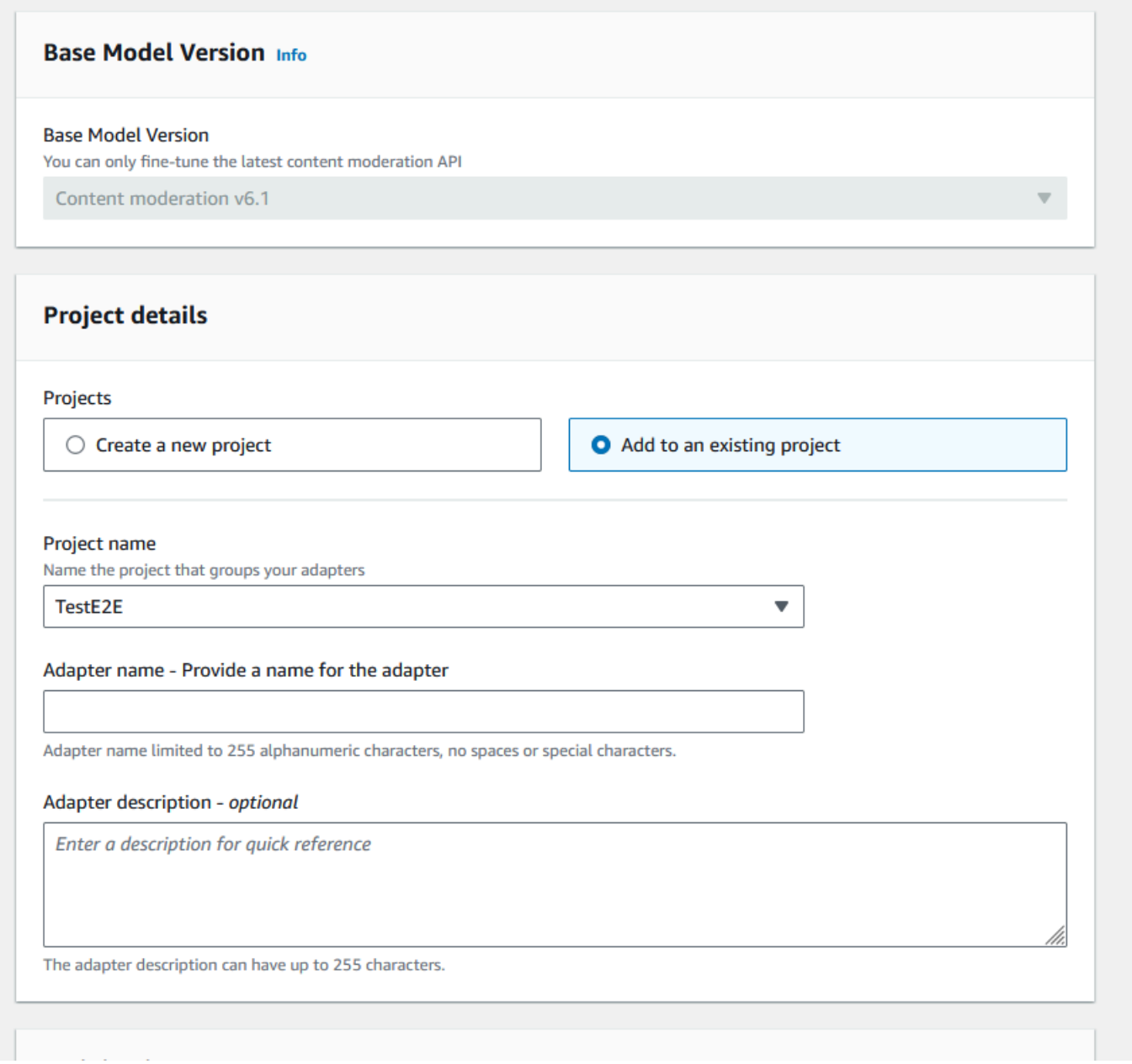


3. Con el panel Umbral, puede cambiar el umbral de confianza mínimo que debe tener el adaptador para asignar una etiqueta a una imagen. El número de instancias marcadas y no marcadas cambiará a medida que ajuste el umbral de confianza. También puede filtrar por categoría de etiqueta para ver las métricas de las categorías que ha seleccionado. Establezca el umbral que ha elegido.
4. Puede evaluar el rendimiento del adaptador a partir de los datos de prueba examinando las métricas del panel Rendimiento del adaptador. Estas métricas se calculan comparando las extracciones del adaptador con las anotaciones de «verdad fundamental» del conjunto de pruebas.

El panel de rendimiento del adaptador muestra las tasas de mejora de falsos positivos y falsos negativos del adaptador que ha creado. La pestaña Rendimiento por etiqueta se puede utilizar para comparar el rendimiento del adaptador y del modelo base en cada categoría de etiquetas. Muestra los recuentos de predicciones de falsos positivos y falsos negativos tanto del modelo base como del adaptador, estratificados por categoría de etiqueta. Al revisar estas métricas, puede determinar en qué aspectos es necesario mejorar el adaptador. Para obtener más información sobre estas métricas, consulte Evaluación y mejora del adaptador.

Para mejorar el rendimiento, puede recopilar más imágenes de entrenamiento y, a continuación, crear un nuevo adaptador basado en el proyecto. Solo tiene que volver a la página de inicio de la moderación personalizada y crear un nuevo adaptador dentro de su proyecto, con más imágenes de entrenamiento para que pueda entrenar el adaptador. Esta vez, elija la opción Añadir a un proyecto existente en lugar de Crear un proyecto nuevo y seleccione el proyecto en el que quiere crear el

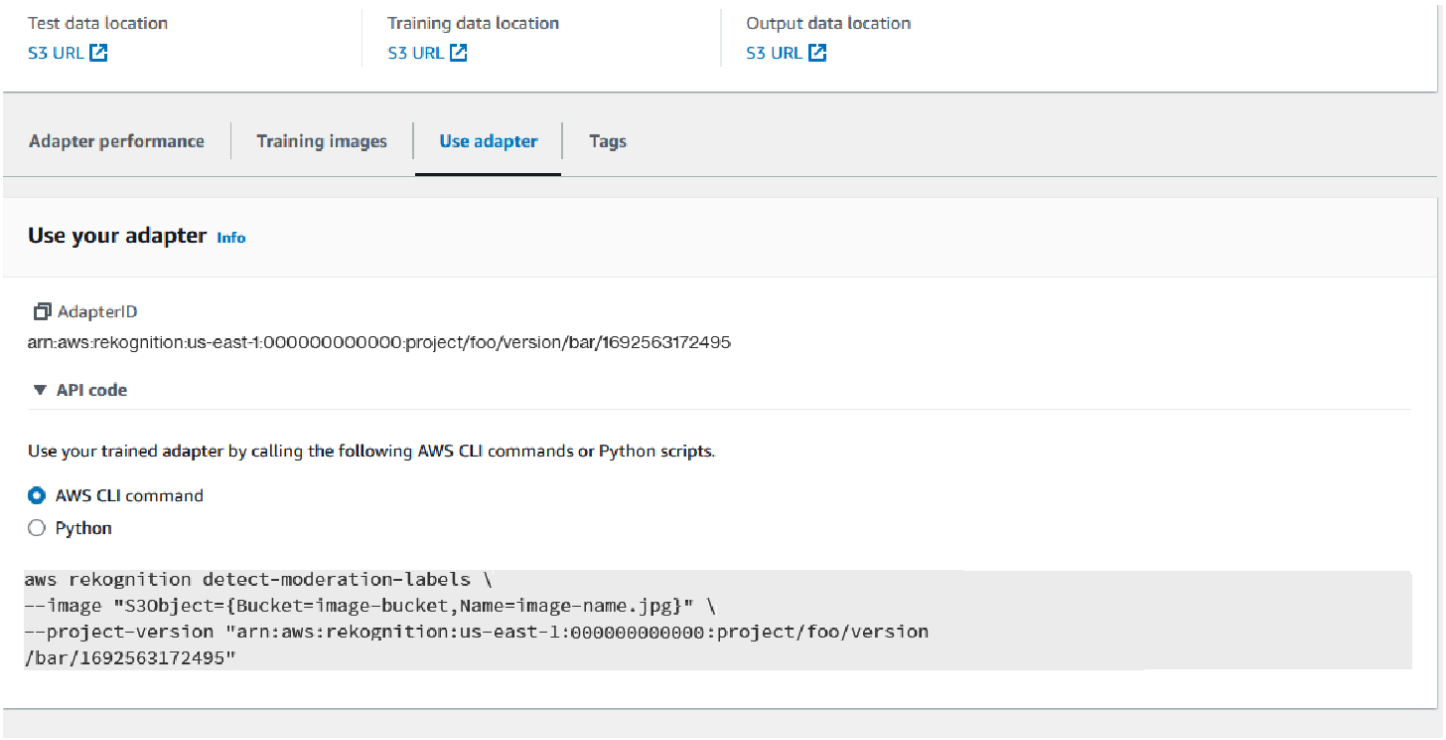
nuevo adaptador en el menú desplegable Nombre del proyecto. Como antes, anote sus imágenes o proporcione un archivo de manifiesto con anotaciones.



Uso del adaptador

Una vez que haya creado su adaptador, puede suministrarlo a una operación de Rekognition compatible, como. DetectModerationLabels Para ver ejemplos de código que puede usar para realizar inferencias con su adaptador, seleccione la pestaña «Usar adaptador», donde podrá ver ejemplos de código tanto para la AWS CLI como para Python. También puede visitar la sección de

la documentación correspondiente a la operación para la que ha creado un adaptador para ver más ejemplos de código, instrucciones de configuración y un ejemplo de JSON.



Eliminar el adaptador y el proyecto

Puede eliminar adaptadores individuales o eliminar su proyecto. Debe eliminar cada adaptador asociado al proyecto para poder eliminar el proyecto en sí.

1. Para eliminar un adaptador asociado al proyecto, elija el adaptador y, a continuación, elija Eliminar.
2. Para eliminar un proyecto, elija el proyecto que desee eliminar y, a continuación, elija Eliminar.

Evaluación y mejora del adaptador

Después de cada ronda de entrenamiento de adaptadores, querrá revisar las métricas de rendimiento de la consola de Rekognition para determinar cuánto se acerca el adaptador al nivel de rendimiento deseado. Luego, puede mejorar aún más la precisión del adaptador para sus imágenes cargando un nuevo lote de imágenes de entrenamiento y entrenando un nuevo adaptador dentro de su proyecto. Una vez que haya creado una versión mejorada del adaptador, puede utilizar la consola para eliminar las versiones anteriores del adaptador que ya no necesite.

También puedes recuperar métricas mediante la operación de la DescribeProjectVersionsAPI.

Métricas de desempeño

Una vez que haya terminado el proceso de entrenamiento y creado el adaptador, es importante evaluar cómo de bien extrae el adaptador la información de las imágenes.

En la consola de Rekognition se proporcionan dos métricas para ayudarle a analizar el rendimiento de su adaptador: mejora con falsos positivos y mejora con falsos negativos.

Puede ver estas métricas de cualquier adaptador seleccionando la pestaña Rendimiento del adaptador en la parte del adaptador de la consola. El panel de rendimiento del adaptador muestra las tasas de mejora de falsos positivos y falsos negativos del adaptador que ha creado.

La mejora de falsos positivos mide cuánto ha mejorado el reconocimiento de falsos positivos por parte del adaptador con respecto al modelo base. Si el valor de mejora de falsos positivos es del 25 %, significa que el adaptador mejoró su reconocimiento de falsos positivos en un 25 % en el conjunto de datos de prueba.

La mejora de falsos negativos mide cuánto ha mejorado el reconocimiento de falsos negativos por parte del adaptador con respecto al modelo base. Si el valor de mejora de falsos negativos es del 25 %, significa que el adaptador mejoró su reconocimiento de falsos negativos en un 25 % en el conjunto de datos de prueba.

La pestaña Rendimiento por etiqueta se puede utilizar para comparar el rendimiento del adaptador y del modelo base en cada categoría de etiquetas. Muestra los recuentos de predicciones de falsos positivos y falsos negativos tanto del modelo base como del adaptador, estratificados por categoría de etiqueta. Al revisar estas métricas, puede determinar en qué aspectos es necesario mejorar el adaptador.

Por ejemplo, si la tasa de falsos negativos del modelo base para la categoría de etiquetas con contenido alcohólico es 15, mientras que la tasa de falsos negativos del adaptador es 15 o superior, sabe que debe centrarse en añadir más imágenes que contengan la etiqueta de alcohol al crear un adaptador nuevo.

Cuando se utilizan las operaciones de la API Rekognition, se devuelve la métrica F1-Score al llamar a la operaciónn. DescribeProjectVersions

Mejorar su modelo

La implementación del adaptador es un proceso iterativo, ya que es probable que necesite entrenarlo varias veces para alcanzar el nivel de precisión deseado. Después de crear y entrenar el adaptador, querrá probar y evaluar su rendimiento en varios tipos de etiquetas.

Si la precisión del adaptador es deficiente en alguna zona, añada nuevos ejemplos de esas imágenes para aumentar el rendimiento del adaptador para esas etiquetas. Intente proporcionarle al adaptador ejemplos adicionales y variados que reflejen los casos en los que tiene dificultades. Al proporcionarle al adaptador imágenes representativas y variadas, podrá gestionar diversos ejemplos del mundo real.

Después de añadir nuevas imágenes al conjunto de entrenamiento, vuelva a entrenar el adaptador y, a continuación, vuelva a evaluarlo en el conjunto de prueba y en las etiquetas. Repita este proceso hasta que el adaptador alcance el nivel de rendimiento deseado. Si proporciona imágenes y anotaciones más representativas, las puntuaciones de falsos positivos y falsos negativos mejorarán gradualmente a lo largo de las sucesivas iteraciones de entrenamiento.

Formatos de archivo de manifiesto

En las siguientes secciones se muestran ejemplos de los formatos de los archivos de manifiesto para los archivos de entrada, salida y evaluación.

Manifiesto de entrada

Un archivo de manifiesto es un archivo delimitado por líneas de JSON, en el que cada línea contiene un JSON que contiene información sobre una sola imagen.

Cada entrada del manifiesto de entrada debe contener el campo `source-ref` con una ruta a la imagen en el bucket de Amazon S3 y, en el caso de la moderación personalizada, el campo `content-moderation-groundtruth` con anotaciones básicas. Se espera que todas las imágenes de un conjunto de datos se encuentren en el mismo bucket. La estructura es común a los archivos de manifiesto de entrenamiento y prueba.

La operación `CreateProjectVersion`de Moderación personalizada utiliza la información proporcionada en el manifiesto de entrada para entrenar un adaptador.

El siguiente ejemplo es una línea de un archivo de manifiesto para una sola imagen que contiene una sola clase insegura:

```
{
   "source-ref": "s3://foo/bar/1.jpg",
   "content-moderation-groundtruth": {
        "ModerationLabels": [
            {
                "Name": "Rude Gesture"
            }
```

```
        ]
    }
}
```

El siguiente ejemplo es una línea de un archivo de manifiesto para una imagen única e insegura que contiene varias clases no seguras, específicamente Desnudo y Gesto grosero.

```
{
    "source-ref": "s3://foo/bar/1.jpg",
    "content-moderation-groundtruth": {
         "ModerationLabels": [
             {
                 "Name": "Rude Gesture"
             },
             {
                 "Name": "Nudity"
             }
         ]
    }
}
```

El siguiente ejemplo es una línea de un archivo de manifiesto para una sola imagen que no contiene ninguna clase no segura:

```
{
    "source-ref": "s3://foo/bar/1.jpg",
    "content-moderation-groundtruth": {
         "ModerationLabels": []
    }
}
```

Para ver la lista completa de etiquetas compatibles, consulte Moderar el contenido.

Manifiesto de salida

Al finalizar un trabajo de entrenamiento, se devuelve un archivo de manifiesto de salida. El archivo de manifiesto de salida es un archivo delimitado por líneas de JSON, en el que cada línea contiene un JSON con la información de una sola imagen. La ruta de Amazon S3 al se OutputManifest puede obtener a partir de la `DescribeProjectVersion` respuesta:

- `TrainingDataResult.Output.Assets[0].GroundTruthManifest.S3Object` para el conjunto de datos de entrenamiento
- `TestingDataResult.Output.Assets[0].GroundTruthManifest.S3Object` para el conjunto de datos de pruebas

Se devuelve la siguiente información para cada entrada del manifiesto de salida:

Nombre de clave	Description (Descripción)
`source-ref`	Referencia a una imagen de S3 que se proporcionó en el manifiesto de entrada
`content-moderation-groundtruth`	Fundamenta las anotaciones veraces que se proporcionaron en el manifiesto de entrada
`detect-moderation-labels`	Las predicciones del adaptador, solo forman parte del conjunto de datos de prueba
`detect-moderation-labels-base-model`	Predicciones del modelo base, solo parte del conjunto de datos de prueba

Las predicciones del adaptador y del modelo base se devuelven en ConfidenceTrehsold 5.0 en un formato similar al de la DetectModerationLabelsrespuesta.

El siguiente ejemplo muestra la estructura de las predicciones del modelo base y del adaptador:

```
{
   "ModerationLabels": [
      {
         "Confidence": number,
         "Name": "string",
         "ParentName": "string"
      }
   ],
   "ModerationModelVersion": "string",
   "ProjectVersion": "string"
}
```

Para ver la lista completa de etiquetas devueltas, consulte Moderar el contenido.

Manifiesto de resultados de evaluación

Al finalizar un trabajo de entrenamiento, se devuelve un archivo con el manifiesto de los resultados de la evaluación. El manifiesto de los resultados de la evaluación es un archivo JSON generado por el trabajo de entrenamiento y contiene información sobre el rendimiento del adaptador con los datos de la prueba.

La ruta de Amazon S3 al manifiesto de los resultados de la evaluación se puede obtener en el `EvaluationResult.Summary.S3Object` campo de la DescribeProejctVersion respuesta.

La estructura del manifiesto de resultados de evaluación se muestra en el ejemplo siguiente:

```
{
    "AggregatedEvaluationResults": {
       "F1Score": number
    },

    "EvaluationDetails": {
        "EvaluationEndTimestamp": "datetime",
        "Labels": [
            "string"
        ],
        "NumberOfTestingImages": number,
        "NumberOfTrainingImages": number,
        "ProjectVersionArn": "string"
    },

    "ContentModeration": {
        "InputConfidenceThresholdEvalResults": {
            "ConfidenceThreshold": float,
            "AggregatedEvaluationResults": {
                "BaseModel": {
                    "TruePositive": int,
                    "TrueNegative": int,
                    "FalsePositive": int,
                    "FalseNegative": int
                },
                "Adapter": {
                    "TruePositive": int,
                    "TrueNegative": int,
                    "FalsePositive": int,
                    "FalseNegative": int
                }
```

```
            },
            "LabelEvaluationResults": [
                {
                    "Label": "string",
                    "BaseModel": {
                        "TruePositive": int,
                        "TrueNegative": int,
                        "FalsePositive": int,
                        "FalseNegative": int
                    },
                    "Adapter": {
                        "TruePositive": int,
                        "TrueNegative": int,
                        "FalsePositive": int,
                        "FalseNegative": int
                    }
                }
            ]
        }
        "AllConfidenceThresholdsEvalResults": [
            {
                "ConfidenceThreshold": float,
                "AggregatedEvaluationResults": {
                    "BaseModel": {
                        "TruePositive": int,
                        "TrueNegative": int,
                        "FalsePositive": int,
                        "FalseNegative": int
                    },
                    "Adapter": {
                        "TruePositive": int,
                        "TrueNegative": int,
                        "FalsePositive": int,
                        "FalseNegative": int
                    }
                },
                "LabelEvaluationResults": [
                    {
                       "Label": "string",
                        "BaseModel": {
                            "TruePositive": int,
                            "TrueNegative": int,
                            "FalsePositive": int,
                            "FalseNegative": int
```

```
                        },
                        "Adapter": {
                            "TruePositive": int,
                            "TrueNegative": int,
                            "FalsePositive": int,
                            "FalseNegative": int
                        }
                    }
                ]
            }
        ]
    }
}
```

El archivo del manifiesto de evaluación contiene:

- Resultados agregados según lo definido por `F1Score`
- Detalles del trabajo de evaluación ProjectVersionArn, incluidos el número de imágenes de entrenamiento, el número de imágenes de prueba y las etiquetas con las que se entrenó el adaptador.
- FalseNegative Resultados agregados TruePositive y obtenidos tanto para el modelo base como para el rendimiento del adaptador. TrueNegative FalsePositive
- Por etiqueta TruePositive TrueNegative FalsePositive, y FalseNegative resultados tanto para el modelo base como para el rendimiento del adaptador, calculados según el umbral de confianza de entrada.
- Resultados agregados y por etiqueta TruePositive TrueNegative FalsePositive, y FalseNegative resultados para el rendimiento del modelo base y del adaptador con diferentes umbrales de confianza. El umbral de confianza oscila entre 5 y 100 en pasos de 5.

Prácticas recomendadas de adaptadores de entrenamiento

Se sugiere que siga las siguientes prácticas recomendadas al crear, entrenar y usar sus adaptadores:

1. Los datos de imagen de muestra deben capturar los errores representativos que los clientes pretenden suprimir. Si el modelo comete errores repetidos en imágenes visualmente similares, asegúrese de incluir muchas de esas imágenes para el entrenamiento.
2. En lugar de incluir únicamente imágenes en las que el modelo comete errores en una etiqueta de moderación concreta, asegúrese también de incluir imágenes en las que el modelo no esté cometiendo errores en esa etiqueta de moderación.
3. Proporcione un mínimo de cincuenta muestras de falsos negativos o veinte muestras de falsos positivos para el entrenamiento y un mínimo de veinte muestras para realizar pruebas. Sin embargo, suministre tantas imágenes anotadas como sea posible para mejorar el rendimiento del adaptador.
4. Anotar todas las etiquetas que le interesan para todas las imágenes: si decide que necesita anotar la aparición de una etiqueta en una imagen, asegúrese de anotar la aparición de esta etiqueta en todas las demás imágenes.
5. Los datos de la imagen de muestra deben contener tantas variaciones en la etiqueta como sea posible, centrándose en las instancias que sean representativas de las imágenes que se analizarán en un entorno de producción.

Configuración de permisos AutoUpdate

Rekognition admite AutoUpdate la función de adaptadores personalizados. Esto significa que el reentrenamiento automatizado se hace todo lo posible cuando el AutoUpdate indicador está ACTIVADO en un proyecto. Estas actualizaciones automáticas requieren permiso para acceder a sus Training/Testing conjuntos de datos y a la AWS KMS clave con la que entrena a su adaptador de cliente. Puede proporcionar estos permisos siguiendo los pasos que se indican a continuación.

Permisos de bucket de Amazon S3

De forma predeterminada, todos los buckets y objetos de Amazon S3 son privados. Solo el propietario del recurso, la AWS cuenta que creó el depósito, puede acceder al depósito y a cualquier objeto que contenga. No obstante, el propietario del recurso puede elegir conceder permisos de acceso a otros recursos y usuarios escribiendo una política de bucket.

Si desea crear o modificar un bucket de Amazon S3 para que se utilice como fuente de conjuntos de datos de entrada y destino de los resultados de entrenamiento, deberá modificar aún más la política del bucket. Para leer o escribir en un bucket de Amazon S3, Rekognition debe tener los siguientes permisos.

Política de Amazon S3 requerida por Rekognition

Rekognition requiere una política de permisos con los siguientes atributos:

- El SID de la instrucción
- El nombre del bucket
- El nombre de la entidad principal del servicio para Rekognition.
- Los recursos necesarios para Rekognition, el bucket y todo su contenido
- Las acciones necesarias que Rekognition debe realizar.

La siguiente política permite que Rekognition acceda a un bucket de Amazon S3 durante el reentrenamiento automático.

```
{
    "Statement": [
        {
            "Effect": "Allow",
            "Sid": "AllowRekognitionAutoUpdateActions",
            "Principal": {
                "Service": "rekognition.amazonaws.com"
            },
            "Action": [
                "s3:ListBucket",
                "s3:GetObject",
                "s3:PutObject",
                "s3:HeadObject",
                "s3:HeadBucket"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket",
                "arn:aws:s3:::amzn-s3-demo-bucket/*"
            ]
        }
    ]
}
```

Puede seguir esta guía para añadir la política de bucket anterior a su bucket de S3.

Consulte más información sobre las políticas de bucket aquí.

AWS KMS Permisos clave

Rekognition te permite proporcionar un adaptador personalizado KmsKeyId opcional mientras entrenas. Cuando se proporciona, Rekognition usa esta clave para cifrar las imágenes de entrenamiento y prueba copiadas en el servicio para el entrenamiento de modelos. La clave también se usa para cifrar los resultados del entrenamiento y los archivos de manifiesto escritos en el bucket de salida de Amazon S3 (OutputConfig).

Si decide proporcionar una clave de KMS como entrada para su formación de adaptadores personalizada (es decir,`Rekognition:CreateProjectVersion`), debe modificar aún más la política de claves de KMS para permitir que la entidad principal del servicio de Rekognition utilice esta clave para el reentrenamiento automatizado en el futuro. Rekognition debe tener los siguientes permisos.

Política clave requerida de Rekognition AWS KMS

Amazon Rekognition requiere una política de permisos con los siguientes atributos:

- El SID de la instrucción
- El nombre de la entidad principal del servicio para Amazon Rekognition.
- Las acciones necesarias que Amazon Rekognition debe realizar.

La siguiente política de claves permite a Amazon Rekognition acceder a una clave de Amazon KMS durante el reentrenamiento automatizado:

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "KeyPermissions",
            "Effect": "Allow",
            "Principal": {
                "Service": "rekognition.amazonaws.com"
            },
            "Action": [
                "kms:DescribeKey",
                "kms:GenerateDataKey",
                "kms:Decrypt"
```

```
            ],
            "Resource": "*"
        }
    ]
}
```

Puede seguir esta guía para añadir la AWS KMS política anterior a su clave. AWS KMS

Consulta más información sobre AWS KMS las políticas aquí.

AWS Notificación de Health Dashboard para Rekognition

Tu AWS Health Dashboard ofrece soporte para las notificaciones que provienen de Rekognition. Estas notificaciones proporcionan orientación sobre el conocimiento y la corrección de los cambios programados en los modelos de Rekognition que puedan afectar a sus aplicaciones. Actualmente solo están disponibles los eventos específicos de la característica Moderación de contenido de Rekognition.

El AWS Health Dashboard forma parte del servicio de AWS Salud. No precisa configuración, y cualquier usuario autenticado en su cuenta puede consultarlo. Para obtener más información, consulte Getting started wtih the AWS Health Dashboard.

Si recibe un mensaje de notificación similar a los siguientes mensajes, debe tratarse como una alarma para tomar medidas.

Ejemplo de notificación: Hay disponible una nueva versión del modelo para Moderación de contenido de Rekognition.

Rekognition publica el `AWS_MODERATION_MODEL_VERSION_UPDATE_NOTIFICATION` evento en el AWS Health Dashboard para indicar que se ha lanzado una nueva versión del modelo de moderación. Este evento es importante si utiliza la DetectModerationLabels API y los adaptadores con esta API. Los nuevos modelos pueden afectar a la calidad en función de su caso de uso y, con el tiempo, sustituirán a las versiones anteriores de los modelos. Se recomienda validar la calidad del modelo y tener en cuenta los plazos de actualización del modelo cuando reciba esta alerta.

Si recibe una notificación de actualización de la versión del modelo, debe tratarla como una alarma para tomar medidas. Si no usa adaptadores, debe evaluar la calidad del modelo actualizado en su caso de uso actual. Si usa adaptadores, debe entrenar a los nuevos adaptadores con el modelo actualizado y evaluar su calidad. Si tiene un conjunto de entrenamiento automático, los nuevos adaptadores se entrenarán automáticamente y, a continuación, podrá evaluar su calidad.

```
{
   "version": "0",
    "id": "id-number",
    "detail-type": "AWS Health Event",
    "source": "aws.health",
    "account": "123456789012",
    "time": "2023-10-06T06:27:57Z",
    "region": "region",
    "resources": [],
    "detail": {
        "eventArn": "arn:aws:health:us-east-1::event/
AWS_MODERATION_MODEL_UPDATE_NOTIFICATION_event-number",
        "service": "Rekognition",
        "eventTypeCode": "AWS_MODERATION_MODEL_VERSION_UPDATE_NOTIFICATION",
        "eventScopeCode": "ACCOUNT_SPECIFIC",
        "communicationId": "communication-id-number",
        "eventTypeCategory": "scheduledChange",
        "startTime": "Fri, 05 Apr 2023 12:00:00 GMT",
        "lastUpdatedTime": "Fri, 05 Apr 2023 12:00:00 GMT",
        "statusCode": "open",
        "eventRegion": "us-east-1",
        "eventDescription": [
            {
                "language": "en_US",
                "latestDescription": "A new model version is available for Rekognition
 Content Moderation."
            }
        ]
    }
}
```

Consulte Monitorear los eventos de salud de AWS con Amazon EventBridge para detectar y reaccionar ante los eventos de AWS salud mediante EventBridge.

Revisión de contenido inapropiado con Amazon Augmented AI

Amazon Augmented AI (Amazon A2I) le permite crear los flujos de trabajo necesarios para la revisión humana de las predicciones de machine learning.

Amazon Rekognition está integrado directamente con Amazon A2I para que pueda implementar fácilmente la revisión humana para el caso de detección de imágenes no seguras. Amazon A2I proporciona un flujo de trabajo de revisión humana para la moderación de imágenes. Esto le permite revisar fácilmente las predicciones de Amazon Rekognition. Puede definir umbrales de confianza para su caso de uso y ajustarlos a lo largo del tiempo. Con Amazon A2I, puede utilizar un grupo de revisores dentro de su propia organización o Amazon Mechanical Turk. Además puede usar proveedores de mano de obra preseleccionados por AWS en función de su calidad y su cumplimiento de los procedimientos de seguridad.

En los siguientes pasos se explica cómo configurar Amazon A2I con Amazon Rekognition. En primer lugar, cree una definición de flujo con Amazon A2I que tenga las condiciones que desencadenan la revisión humana. A continuación, pase el nombre de recurso de Amazon (ARN) de la definición de flujo a la operación `DetectModerationLabel` de Amazon Rekognition. En la respuesta `DetectModerationLabel`, puede ver si se requiere revisión humana. Los resultados de la revisión humana están disponibles en un bucket de Amazon S3 establecido por la definición de flujo.

Para ver una end-to-end demostración de cómo utilizar Amazon A2I con Amazon Rekognition, consulte uno de los siguientes tutoriales de la Guía para desarrolladores de Amazon AI. SageMaker

- Demostración: Introducción a la consola Amazon A2I
- Demostración: Introducción al uso de la API de Amazon A2I

  Para empezar a utilizar la API, también puede ejecutar un ejemplo de cuaderno de Jupyter. Consulte Utilizar una instancia de SageMaker Notebook con Amazon A2I Jupyter Notebook para utilizar la integración de Amazon Augmented AI (Amazon A2I) con Amazon Rekognition [Ejemplo] en una instancia de bloc de notas con IA. SageMaker

Ejecución DetectModerationLabels con Amazon A2I

> **Note**
>
> Cree todos sus recursos de Amazon Rekognition y de Amazon A2I en la misma región de AWS.

1. Complete los requisitos previos que se indican en la sección Introducción a Amazon Augmented AI en la documentación sobre SageMaker IA.

Además, recuerda configurar tus permisos de IAM tal y como se indica en la página Permisos y seguridad en Amazon Augmented AI de la documentación sobre SageMaker IA.

2. Siga las instrucciones para crear un flujo de trabajo de revisión humana que aparecen en la documentación sobre SageMaker IA.

   Un flujo de trabajo de revisión humana gestiona el procesamiento de una imagen. Contiene las condiciones que desencadenan una revisión humana, el equipo de trabajo al que se envía la imagen, la plantilla de interfaz de usuario que utiliza el equipo de trabajo y el bucket de Amazon S3 al que se envían los resultados del equipo de trabajo.

   En tu `CreateFlowDefinition` llamada, tendrás que configurar el botón `HumanLoopRequestSource` en «AWS/Rekognition/DetectModerationLabels/Image/V3". Después de eso, debe decidir cómo desea configurar sus condiciones que desencadenan una revisión humana.

   Con Amazon Rekognition tiene dos opciones para `ConditionType`: `ModerationLabelConfidenceCheck` y `Sampling`.

   `ModerationLabelConfidenceCheck` crea un bucle humano cuando la confianza de una etiqueta de moderación se encuentra dentro de un rango. Por último, `Sampling` envía un porcentaje aleatorio de los documentos procesados para revisión humana. Cada `ConditionType` utiliza un conjunto diferente de `ConditionParameters` para establecer los resultados de la revisión humana.

   `ModerationLabelConfidenceCheck` tiene el `ConditionParameters` `ModerationLableName`, que establece la clave que requiere revisión humana. Además, tiene confianza, que establece el rango porcentual para enviar a revisión humana con LessThan GreaterThan, y Equals. `Samplingestablece` `RandomSamplingPercentage` el porcentaje de documentos que se enviarán a revisión humana.

   El ejemplo de código siguiente es una llamada parcial de `CreateFlowDefinition`. Envía una imagen para revisión humana si tiene una calificación inferior al 98% en la etiqueta «Sugerente», y más del 95% en la etiqueta «Trajes de baño o ropa interior femenina». Esto significa que si la imagen no se considera sugerente pero sí tiene una mujer en ropa interior o en traje de baño, puede volver a comprobar la imagen mediante la revisión humana.

```
    def create_flow_definition():
```

```
    '''
    Creates a Flow Definition resource

    Returns:
    struct: FlowDefinitionArn
    '''
    humanLoopActivationConditions = json.dumps(
        {
            "Conditions": [
                {
                  "And": [
                    {
                        "ConditionType": "ModerationLabelConfidenceCheck",
                        "ConditionParameters": {
                            "ModerationLabelName": "Suggestive",
                            "ConfidenceLessThan": 98
                        }
                    },
                    {
                        "ConditionType": "ModerationLabelConfidenceCheck",
                        "ConditionParameters": {
                            "ModerationLabelName": "Female Swimwear Or Underwear",
                            "ConfidenceGreaterThan": 95
                        }
                    }
                  ]
               }
            ]
        }
    )
```

`CreateFlowDefinition` devuelve un `FlowDefinitionArn`, que utilizará en el siguiente paso cuando llame a `DetectModerationLabels`.

Para obtener más información, consulta CreateFlowDefinitionla referencia de la API de SageMaker IA.

3. Establezca el parámetro `HumanLoopConfig` cuando llame a `DetectModerationLabels`, como en Detección de imágenes inapropiadas. Consulte el paso 4 para ver ejemplos de una llamada de `DetectModerationLabels` con `HumanLoopConfig` establecido.

a. Dentro del parámetro `HumanLoopConfig`, establezca el `FlowDefinitionArn` en el ARN de la definición de flujo que creó en el paso 2.

b. Establezca su `HumanLoopName`. Este debe ser único dentro de cada región y debe estar en minúsculas.

c. (Opcional) Puede usar `DataAttributes` para establecer si la imagen que pasó a Amazon Rekognition está libre de información de identificación personal o no. Debe establecer este parámetro para poder enviar información a Amazon Mechanical Turk.

4. Ejecute `DetectModerationLabels`.

En los siguientes ejemplos se muestra cómo utilizar el `HumanLoopConfig` conjunto AWS CLI y AWS SDK para Python (Boto3) cómo ejecutar `DetectModerationLabels` con él.

AWS SDK para Python (Boto3)

En los siguientes ejemplos de solicitud, se utiliza el SDK para Python (Boto3). Para obtener más información, consulte detect_moderation_labels en la documentación de referencia de la API de AWS SDK para Python (Boto).

```
import boto3

rekognition = boto3.client("rekognition", aws-region)

response = rekognition.detect_moderation_labels( \
        Image={'S3Object': {'Bucket': bucket_name, 'Name': image_name}}, \
        HumanLoopConfig={ \
            'HumanLoopName': 'human_loop_name', \
            'FlowDefinitionArn': , "arn:aws:sagemaker:aws-
region:aws_account_number:flow-definition/flow_def_name" \
            'DataAttributes': {'ContentClassifiers':
 ['FreeOfPersonallyIdentifiableInformation','FreeOfAdultContent']}
         })
```

AWS CLI

El siguiente ejemplo de solicitud utiliza la AWS CLI. Para obtener más información, consulte detect-moderation-labels en la Referencia de comandos de la AWS CLI.

```
$ aws rekognition detect-moderation-labels \
    --image "S3Object={Bucket='bucket_name',Name='image_name'}" \
```

```
    --human-loop-config
 HumanLoopName="human_loop_name",FlowDefinitionArn="arn:aws:sagemaker:aws-
region:aws_account_number:flow-
definition/
flow_def_name",DataAttributes='{ContentClassifiers=["FreeOfPersonallyIdentifiableInforma
 "FreeOfAdultContent"]}'
```

```
$ aws rekognition detect-moderation-labels \
    --image "S3Object={Bucket='bucket_name',Name='image_name'}" \
    --human-loop-config \
        '{"HumanLoopName": "human_loop_name", "FlowDefinitionArn":
 "arn:aws:sagemaker:aws-region:aws_account_number:flow-
definition/flow_def_name", "DataAttributes": {"ContentClassifiers":
 ["FreeOfPersonallyIdentifiableInformation", "FreeOfAdultContent"]}}'
```

Cuando se ejecuta `DetectModerationLabels` con la `HumanLoopConfig` opción activada, Amazon Rekognition SageMaker llama a la operación de la API de IA. `StartHumanLoop` Este comando toma la respuesta de `DetectModerationLabels` y la compara con las condiciones de definición de flujo del ejemplo. Si cumple con las condiciones de revisión, devuelve un `HumanLoopArn`. Esto significa que los miembros del equipo de trabajo que configuró en su definición de flujo ahora pueden revisar la imagen. Llamar a la operación de tiempo de ejecución de la inteligencia artificial aumentada de Amazon `DescribeHumanLoop` proporciona información sobre el resultado del bucle. Para obtener más información, consulte la documentación DescribeHumanLoopde referencia de la API de Amazon Augmented AI.

Después de revisar la imagen, puede ver los resultados en el bucket especificado en la ruta de salida de la definición de flujo. Amazon A2I también le notificará a Amazon CloudWatch Events cuando se complete la revisión. Para ver qué eventos debe buscar, consulte la sección CloudWatch Eventos en la documentación de SageMaker IA.

Para obtener más información, consulte Introducción a Amazon Augmented AI en la documentación de SageMaker IA.

Detección de texto

Amazon Rekognition puede detectar texto en imágenes y vídeos. A continuación, puede convertir el texto detectado en texto legible por una máquina. Puede utilizar la detección de texto legible por una máquina en imágenes para implementar soluciones como:

- Búsqueda visual. Por ejemplo, recuperar y mostrar imágenes que contengan el mismo texto.
- Información de contenido. Por ejemplo, proporcionar información sobre los temas que aparecen en el texto reconocido en fotogramas de vídeo extraídos. La aplicación puede realizar búsquedas de texto reconocido sobre contenido relevante como, por ejemplo, noticias, puntuaciones deportivas, dorsales de deportista y titulares.
- Navegación. Por ejemplo, el desarrollo de una app mediante diálogos para personas con problemas de visión que reconozca los nombres de restaurantes, tiendas o letreros de calles.
- Soporte para seguridad pública y transporte. Por ejemplo, la detección de números de matrícula de imágenes de cámaras de tráfico.
- Filtrado. Por ejemplo, filtrado de información de identificación personal (PII) a partir de imágenes.

Para la detección de texto en vídeos, puede implementar soluciones como:

- Búsqueda de vídeos para clips donde se muestren las palabras clave de texto específicas, como el nombre del invitado en un gráfico en un programa de noticias.
- Moderación del contenido para cumplir con los estándares de la organización mediante la detección de texto accidental, blasfemias o spam.
- Búsqueda de todas las superposiciones de texto en la línea temporal del vídeo para su posterior procesamiento, como la sustitución de texto por texto en otro idioma para la internacionalización del contenido.
- Búsqueda de ubicaciones de texto, de modo que otros gráficos se puedan alinear según corresponda.

Para detectar texto en imágenes en formato JPEG o PNG, utilice la DetectTextoperación.
Para detectar texto en vídeo de forma asíncrona, utilice las StartTextDetectionoperaciones y. GetTextDetection Las operaciones de detección de texto de imagen y de vídeo admiten la mayoría de las fuentes, incluidas las que tienen un estilo muy sofisticado. Después de detectar texto, Amazon

Rekognition crea una representación de palabras y líneas de texto detectadas, muestra la relación entre ellas y le indica dónde está el texto en una imagen o fotograma de vídeo.

Las operaciones `DetectText` y `GetTextDetection` detectan palabras y líneas. Una palabra es uno o más caracteres del alfabeto que no están separados por espacios. `DetectText` puede detectar hasta 100 palabras en una imagen. `GetTextDetection` también puede detectar hasta 100 palabras por fotograma de vídeo.

Una palabra consta de uno o varios caracteres alfabéticos que no están separados por espacios. Amazon Rekognition está diseñado para detectar palabras en inglés, árabe, ruso, alemán, francés, italiano, portugués y español.

Una línea es una cadena de palabras equidistantes. Una línea no es necesariamente una oración completa (los puntos no indican el final de una línea). Por ejemplo, Amazon Rekognition detecta un número de matrícula como una línea. Además, una línea finaliza cuando no hay texto alineado después de ella o cuando existe un hueco grande entre las palabras, en relación con la longitud de las mismas. En función del hueco entre las palabras, Amazon Rekognition podría detectar varias líneas de texto alineado en la misma dirección. Si una frase abarca varias líneas, la operación devuelve varias líneas.

Analice la siguiente imagen.



Los cuadros azules representan información sobre texto detectado y la ubicación del texto que devuelve la operación `DetectText`. En este ejemplo, Amazon Rekognition detecta "IT'S", "MONDAY", "but", "keep" y "Smiling" como palabras. Amazon Rekognition detecta "IT'S", "MONDAY", "but keep" y "Smiling" como palabras. Para detectarlo, el texto debe estar dentro de una orientación de +/- 90° grados con respecto al eje horizontal.

Para ver un ejemplo, consulta Detección de texto en una imagen.

Temas

- Detección de texto en una imagen
- Detección de texto en un vídeo almacenado

Detección de texto en una imagen

Puede facilitar una imagen de entrada como matriz de bytes de imagen (bytes de imagen con codificación base64) o como objeto de Amazon S3. En este procedimiento, carga una imagen JPEG o PNG en su bucket de S3 y especifica el nombre de archivo.

Para detectar texto en una imagen (API)

1. Complete los siguientes requisitos previos si aún no lo ha hecho.

   a. Cree o actualice un usuario con los permisos `AmazonRekognitionFullAccess` y `AmazonS3ReadOnlyAccess`. Para obtener más información, consulte Paso 1: Configurar una cuenta de AWS y crear un usuario.

   b. Instale y configure las y las AWS Command Line Interface . AWS SDKs Para obtener más información, consulte Paso 2: Configure y AWS CLI AWS SDKs.

2. Cargue la imagen que contenga texto en su bucket de S3.

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

3. Consulte los siguientes ejemplos para llamar a la operación `DetectText`.

   Java

   El siguiente código de ejemplo muestra líneas y palabras que se han detectado en una imagen.

Reemplace los valores de `amzn-s3-demo-bucket` y `photo` por los nombre del bucket de S3 y de la imagen que utilizó en el paso 2.

```
//Copyright 2018 Amazon.com, Inc. or its affiliates. All Rights Reserved.
//PDX-License-Identifier: MIT-0 (For details, see https://github.com/awsdocs/
amazon-rekognition-developer-guide/blob/master/LICENSE-SAMPLECODE.)

package aws.example.rekognition.image;
import com.amazonaws.services.rekognition.AmazonRekognition;
import com.amazonaws.services.rekognition.AmazonRekognitionClientBuilder;
import com.amazonaws.services.rekognition.model.AmazonRekognitionException;
import com.amazonaws.services.rekognition.model.Image;
import com.amazonaws.services.rekognition.model.S3Object;
import com.amazonaws.services.rekognition.model.DetectTextRequest;
import com.amazonaws.services.rekognition.model.DetectTextResult;
import com.amazonaws.services.rekognition.model.TextDetection;
import java.util.List;



public class DetectText {

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


      String photo = "inputtext.jpg";
      String bucket = "bucket";

      AmazonRekognition rekognitionClient =
 AmazonRekognitionClientBuilder.defaultClient();



      DetectTextRequest request = new DetectTextRequest()
              .withImage(new Image()
              .withS3Object(new S3Object()
              .withName(photo)
              .withBucket(bucket)));


      try {
         DetectTextResult result = rekognitionClient.detectText(request);
```

```
         List<TextDetection> textDetections = result.getTextDetections();

         System.out.println("Detected lines and words for " + photo);
         for (TextDetection text: textDetections) {

                 System.out.println("Detected: " + text.getDetectedText());
                 System.out.println("Confidence: " +
 text.getConfidence().toString());
                 System.out.println("Id : " + text.getId());
                 System.out.println("Parent Id: " + text.getParentId());
                 System.out.println("Type: " + text.getType());
                 System.out.println();
         }
      } catch(AmazonRekognitionException e) {
         e.printStackTrace();
      }
   }
}
```

Java V2

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

```
/**
*  To run this code example, ensure that you perform the Prerequisites as stated
 in the Amazon Rekognition Guide:
*  https://docs.aws.amazon.com/rekognition/latest/dg/video-analyzing-with-
sqs.html
*
* Also, ensure that set up your development environment, including your
 credentials.
*
* For information, see this documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
*/

//snippet-start:[rekognition.java2.detect_text.import]
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
```

```
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.DetectTextRequest;
import software.amazon.awssdk.services.rekognition.model.Image;
import software.amazon.awssdk.services.rekognition.model.DetectTextResponse;
import software.amazon.awssdk.services.rekognition.model.TextDetection;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;
import java.util.List;
//snippet-end:[rekognition.java2.detect_text.import]

/**
* Before running this Java V2 code example, set up your development environment,
 including your credentials.
*
* For more information, see the following documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
*/
public class DetectTextImage {

 public static void main(String[] args) {

     final String usage = "\n" +
         "Usage: " +
         "   <sourceImage>\n\n" +
         "Where:\n" +
         "   sourceImage - The path to the image that contains text (for
 example, C:\\AWS\\pic1.png). \n\n";

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

     String sourceImage = args[0] ;
     Region region = Region.US_WEST_2;
     RekognitionClient rekClient = RekognitionClient.builder()
         .region(region)
         .credentialsProvider(ProfileCredentialsProvider.create("default"))
         .build();
```

```
    detectTextLabels(rekClient, sourceImage );
    rekClient.close();
}

// snippet-start:[rekognition.java2.detect_text.main]
public static void detectTextLabels(RekognitionClient rekClient, String
sourceImage) {

    try {
        InputStream sourceStream = new FileInputStream(sourceImage);
        SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);
        Image souImage = Image.builder()
            .bytes(sourceBytes)
            .build();

        DetectTextRequest textRequest = DetectTextRequest.builder()
            .image(souImage)
            .build();

        DetectTextResponse textResponse = rekClient.detectText(textRequest);
        List<TextDetection> textCollection = textResponse.textDetections();
        System.out.println("Detected lines and words");
        for (TextDetection text: textCollection) {
            System.out.println("Detected: " + text.detectedText());
            System.out.println("Confidence: " + text.confidence().toString());
            System.out.println("Id : " + text.id());
            System.out.println("Parent Id: " + text.parentId());
            System.out.println("Type: " + text.type());
            System.out.println();
        }

    } catch (RekognitionException | FileNotFoundException e) {
        System.out.println(e.getMessage());
        System.exit(1);
    }
}
// snippet-end:[rekognition.java2.detect_text.main]
```

AWS CLI

Este AWS CLI comando muestra el resultado JSON de la operación `detect-text` CLI.

Reemplace los valores de `amzn-s3-demo-bucket` y `Name` por los nombre del bucket de S3 y de la imagen que utilizó en el paso 2.

Sustituya el valor de `profile_name` de por el nombre de su perfil de desarrollador.

```
aws rekognition detect-text  --image "{"S3Object":{"Bucket":"amzn-s3-demo-
bucket","Name":"image-name"}}" --profile default
```

Si accede a la CLI en un dispositivo Windows, utilice comillas dobles en lugar de comillas simples y evite las comillas dobles internas con una barra invertida (es decir, \) para corregir cualquier error del analizador que pueda encontrar. Por ver un ejemplo, consulte lo siguiente:

```
aws rekognition detect-text  --image "{\"S3Object\":{\"Bucket\":\"amzn-s3-demo-
bucket\",\"Name\":\"image-name\"}}" --profile default
```

Python

El siguiente código de ejemplo muestra líneas y palabras que se han detectado en una imagen.

Reemplace los valores de `amzn-s3-demo-bucket` y `photo` por los nombre del bucket de S3 y de la imagen que utilizó en el paso 2. Sustituya el valor de `profile_name` en la línea que crea la sesión de Rekognition por el nombre de su perfil de desarrollador.

```
# Copyright 2018 Amazon.com, Inc. or its affiliates. All Rights Reserved.
# PDX-License-Identifier: MIT-0 (For details, see https://github.com/awsdocs/
amazon-rekognition-developer-guide/blob/master/LICENSE-SAMPLECODE.)

import boto3

def detect_text(photo, bucket):

    session = boto3.Session(profile_name='default')
    client = session.client('rekognition')

    response = client.detect_text(Image={'S3Object': {'Bucket': bucket, 'Name':
 photo}})

    textDetections = response['TextDetections']
    print('Detected text\n----------')
```

```
    for text in textDetections:
        print('Detected text:' + text['DetectedText'])
        print('Confidence: ' + "{:.2f}".format(text['Confidence']) + "%")
        print('Id: {}'.format(text['Id']))
        if 'ParentId' in text:
            print('Parent Id: {}'.format(text['ParentId']))
        print('Type:' + text['Type'])
        print()
    return len(textDetections)

def main():
    bucket = 'amzn-s3-demo-bucket'
    photo = 'photo-name'
    text_count = detect_text(photo, bucket)
    print("Text detected: " + str(text_count))

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

.NET

El siguiente código de ejemplo muestra líneas y palabras que se han detectado en una imagen.

Reemplace los valores de `amzn-s3-demo-bucket` y `photo` por los nombre del bucket de S3 y de la imagen que utilizó en el paso 2.

```
//Copyright 2018 Amazon.com, Inc. or its affiliates. All Rights Reserved.
//PDX-License-Identifier: MIT-0 (For details, see https://github.com/awsdocs/
amazon-rekognition-developer-guide/blob/master/LICENSE-SAMPLECODE.)

using System;
using Amazon.Rekognition;
using Amazon.Rekognition.Model;

public class DetectText
{
    public static void Example()
    {
        String photo = "input.jpg";
        String bucket = "amzn-s3-demo-bucket";
```

```
        AmazonRekognitionClient rekognitionClient = new
 AmazonRekognitionClient();

        DetectTextRequest detectTextRequest = new DetectTextRequest()
        {
            Image = new Image()
            {
                S3Object = new S3Object()
                {
                    Name = photo,
                    Bucket = bucket
                }
            }
        };

        try
        {
            DetectTextResponse detectTextResponse =
 rekognitionClient.DetectText(detectTextRequest);
            Console.WriteLine("Detected lines and words for " + photo);
            foreach (TextDetection text in detectTextResponse.TextDetections)
            {
                Console.WriteLine("Detected: " + text.DetectedText);
                Console.WriteLine("Confidence: " + text.Confidence);
                Console.WriteLine("Id : " + text.Id);
                Console.WriteLine("Parent Id: " + text.ParentId);
                Console.WriteLine("Type: " + text.Type);
            }
        }
        catch (Exception e)
        {
            Console.WriteLine(e.Message);
        }
    }
}
```

Node.JS

El siguiente código de ejemplo muestra líneas y palabras que se han detectado en una imagen.

Reemplace los valores de `amzn-s3-demo-bucket` y `photo` por los nombre del bucket de S3 y de la imagen que utilizó en el paso 2. Sustituya el valor de `region` por la región que

aparece en sus credenciales de .aws. Sustituya el valor de `profile_name` en la línea que crea la sesión de Rekognition por el nombre de su perfil de desarrollador.

```
var AWS = require('aws-sdk');

const bucket = 'bucket' // the bucketname without s3://
const photo  = 'photo' // the name of file

const config = new AWS.Config({
  accessKeyId: process.env.AWS_ACCESS_KEY_ID,
  secretAccessKey: process.env.AWS_SECRET_ACCESS_KEY,
})
AWS.config.update({region:'region'});
const client = new AWS.Rekognition();
const params = {
  Image: {
    S3Object: {
      Bucket: bucket,
      Name: photo
    },
  },
}
client.detectText(params, function(err, response) {
  if (err) {
    console.log(err, err.stack); // handle error if an error occurred
  } else {
    console.log(`Detected Text for: ${photo}`)
    console.log(response)
    response.TextDetections.forEach(label => {
      console.log(`Detected Text: ${label.DetectedText}`),
      console.log(`Type: ${label.Type}`),
      console.log(`ID: ${label.Id}`),
      console.log(`Parent ID: ${label.ParentId}`),
      console.log(`Confidence: ${label.Confidence}`),
      console.log(`Polygon: `)
      console.log(label.Geometry.Polygon)
    }
    )
  }
});
```

DetectText solicitud de operación

En la operación `DetectText`, indique una imagen de entrada como matriz de bytes codificada en base64 o como una imagen almacenada en un bucket de Amazon S3. En la siguiente solicitud JSON de ejemplo, aparece la imagen cargada desde un bucket de Amazon S3.

```
{
    "Image": {
        "S3Object": {
            "Bucket": "amzn-s3-demo-bucket",
            "Name": "inputtext.jpg"
        }
    }
}
```

Filtros

El filtrado por región, tamaño y puntuación de confianza del texto le brinda una flexibilidad adicional para controlar el resultado de su detección del texto. Mediante el uso de regiones de interés, puede limitar fácilmente la detección de texto a las regiones que son relevantes para usted, por ejemplo, la parte superior derecha de la foto de perfil o una ubicación fija en relación con un punto de referencia al leer los números de pieza de la imagen de una máquina. El filtro de tamaño del cuadro delimitador de texto puede utilizarse para evitar texto en segundo plano pequeño que sea molesto o irrelevante. El filtro de confianza del texto le permite eliminar resultados que no sean fiables, como texto borroso o difuminado.

Para obtener información sobre los valores de filtro, consulte `DetectTextFilters`.

Puede utilizar los siguientes filtros:

- MinConfidence—Establece el nivel de confianza de la detección de palabras. Las palabras con confianza de detección por debajo de este nivel se excluyen del resultado. Los valores deben estar comprendidos entre 0 y 100.
- MinBoundingBoxWidth— Establece el ancho mínimo del cuadro delimitador de palabras. Las palabras con cuadros delimitadores menores que este valor se excluyen del resultado. El valor es relativo al ancho del marco de la imagen.
- MinBoundingBoxHeight— Establece la altura mínima del cuadro delimitador de palabras. Las palabras con alturas de cuadro delimitador inferiores a este valor se excluyen del resultado. El valor es relativo a la altura del marco de la imagen.

- RegionsOfInterest— Limita la detección a una región específica del marco de la imagen. Los valores son relativos a las dimensiones del marco. Para el texto que se encuentra solo parcialmente dentro de una región, la respuesta es indefinida.

DetectText respuesta de operación

La `DetectText` operación analiza la imagen y devuelve una matriz TextDetections, donde cada elemento (`TextDetection`) representa una línea o palabra detectada en la imagen. Para cada elemento, `DetectText` devuelve la siguiente información:

- El texto detectado (`DetectedText`)
- Las relaciones entre palabras y líneas (`Id` y `ParentId`)
- La ubicación de texto en la imagen (`Geometry`)
- La confianza que Amazon Rekognition tiene en la precisión del texto detectado y cuadro delimitador (`Confidence`)
- El tipo de texto detectado (`Type`)

Texto detectado

Cada elemento `TextDetection` contiene texto reconocido (palabras o líneas) en el campo `DetectedText`. Una palabra consta de uno o varios caracteres alfabéticos que no están separados por espacios. `DetectText` puede detectar hasta 100 palabras en una imagen. El texto devuelto podría incluir caracteres que hacen que una palabra sea irreconocible. Por ejemplo, C@l en lugar de cal. Para determinar si un elemento `TextDetection` representa una línea de texto o una palabra, utilice el campo `Type`.

Cada elemento `TextDetection` incluye un valor de porcentaje que representa el grado de confianza que tiene Amazon Rekognition en la precisión del texto detectado y el cuadro delimitador que rodea el texto.

Relaciones entre palabras y líneas

Cada elemento `TextDetection` tiene un identificador de campo, `Id`. El `Id` muestra la posición de la palabra en una línea. Si el elemento es una palabra, el campo de identificador principal, `ParentId`, identifica la línea en la que se detectó la palabra. El `ParentId` para una línea es nulo. Por ejemplo, la línea "but keep" de la imagen anterior tiene los siguientes valores `Id` y `ParentId`:

Texto	ID	ID principal
but keep	3	
but	8	3
keep	9	3

Ubicación de texto en una imagen

Para determinar dónde está el texto reconocido en una imagen, utilice la información de cuadro delimitador (Geometry) devuelta por `DetectText`. El objeto `Geometry` contiene dos tipos de información de cuadro delimitador para líneas y palabras detectadas:

- Un contorno rectangular amplio alineado con el eje en un objeto BoundingBox
- Un polígono más detallado compuesto por varias coordenadas X e Y en una matriz Point

Las coordenadas del polígono y el cuadro delimitador muestran dónde se encuentra el texto en la imagen de origen. Los valores de las coordenadas son una proporción del tamaño de la imagen global. Para obtener más información, consulte BoundingBox.

La siguiente respuesta JSON de la operación `DetectText` muestra las palabras y las líneas detectadas en la imagen siguiente.




```
{
 'TextDetections': [{'Confidence': 99.35693359375,
                     'DetectedText': "IT'S",
                     'Geometry': {'BoundingBox': {'Height': 0.09988046437501907,
                                                  'Left': 0.6684935688972473,
                                                  'Top': 0.18226495385169983,
                                                  'Width': 0.1461552083492279},
                                  'Polygon': [{'X': 0.6684935688972473,
                                               'Y': 0.1838926374912262},
                                              {'X': 0.8141663074493408,
                                               'Y': 0.18226495385169983},
                                              {'X': 0.8146487474441528,
                                               'Y': 0.28051772713661194},
                                              {'X': 0.6689760088920593,
                                               'Y': 0.2821454107761383}]},
                     'Id': 0,
                     'Type': 'LINE'},
                    {'Confidence': 99.6207275390625,
                     'DetectedText': 'MONDAY',
                     'Geometry': {'BoundingBox': {'Height': 0.11442459374666214,
                                                  'Left': 0.5566731691360474,
```

```
                                        'Top': 0.3525116443634033,
                                        'Width': 0.39574965834617615},
                             'Polygon': [{'X': 0.5566731691360474,
                                          'Y': 0.353712260723114},
                                         {'X': 0.9522717595100403,
                                          'Y': 0.3525116443634033},
                                         {'X': 0.9524227976799011,
                                          'Y': 0.4657355844974518},
                                         {'X': 0.5568241477012634,
                                          'Y': 0.46693623065948486}]},
                            'Id': 1,
                            'Type': 'LINE'},
                           {'Confidence': 99.6160888671875,
                            'DetectedText': 'but keep',
                            'Geometry': {'BoundingBox': {'Height': 0.08314694464206696,
                                                         'Left': 0.6398131847381592,
                                                         'Top': 0.5267938375473022,
                                                         'Width': 0.2021435648202896},
                                         'Polygon': [{'X': 0.640289306640625,
                                                      'Y': 0.5267938375473022},
                                                     {'X': 0.8419567942619324,
                                                      'Y': 0.5295097827911377},
                                                     {'X': 0.8414806723594666,
                                                      'Y': 0.609940767288208},
                                                     {'X': 0.6398131847381592,
                                                      'Y': 0.6072247624397278}]},
                            'Id': 2,
                            'Type': 'LINE'},
                           {'Confidence': 88.95134735107422,
                            'DetectedText': 'Smiling',
                            'Geometry': {'BoundingBox': {'Height': 0.4326171875,
                                                         'Left': 0.46289217472076416,
                                                         'Top': 0.5634765625,
                                                         'Width': 0.5371078252792358},
                                         'Polygon': [{'X': 0.46289217472076416,
                                                      'Y': 0.5634765625},
                                                     {'X': 1.0, 'Y': 0.5634765625},
                                                     {'X': 1.0, 'Y': 0.99609375},
                                                     {'X': 0.46289217472076416,
                                                      'Y': 0.99609375}]},
                            'Id': 3,
                            'Type': 'LINE'},
                           {'Confidence': 99.35693359375,
                            'DetectedText': "IT'S",
```

```
                  'Geometry': {'BoundingBox': {'Height': 0.09988046437501907,
                                               'Left': 0.6684935688972473,
                                               'Top': 0.18226495385169983,
                                               'Width': 0.1461552083492279},
                               'Polygon': [{'X': 0.6684935688972473,
                                            'Y': 0.1838926374912262},
                                           {'X': 0.8141663074493408,
                                            'Y': 0.18226495385169983},
                                           {'X': 0.8146487474441528,
                                            'Y': 0.28051772713661194},
                                           {'X': 0.6689760088920593,
                                            'Y': 0.2821454107761383}]},
                  'Id': 4,
                  'ParentId': 0,
                  'Type': 'WORD'},
                 {'Confidence': 99.6207275390625,
                  'DetectedText': 'MONDAY',
                  'Geometry': {'BoundingBox': {'Height': 0.11442466825246811,
                                               'Left': 0.5566731691360474,
                                               'Top': 0.35251158475875854,
                                               'Width': 0.39574965834617615},
                               'Polygon': [{'X': 0.5566731691360474,
                                            'Y': 0.3537122905254364},
                                           {'X': 0.9522718787193298,
                                            'Y': 0.35251158475875854},
                                           {'X': 0.9524227976799011,
                                            'Y': 0.4657355546951294},
                                           {'X': 0.5568241477012634,
                                            'Y': 0.46693626046180725}]},
                  'Id': 5,
                  'ParentId': 1,
                  'Type': 'WORD'},
                 {'Confidence': 99.96778869628906,
                  'DetectedText': 'but',
                  'Geometry': {'BoundingBox': {'Height': 0.0625,
                                               'Left': 0.6402802467346191,
                                               'Top': 0.5283203125,
                                               'Width': 0.08027780801057816},
                               'Polygon': [{'X': 0.6402802467346191,
                                            'Y': 0.5283203125},
                                           {'X': 0.7205580472946167,
                                            'Y': 0.5283203125},
                                           {'X': 0.7205580472946167,
                                            'Y': 0.5908203125},
```

```
                                        {'X': 0.6402802467346191,
                                         'Y': 0.5908203125}]},
                    'Id': 6,
                    'ParentId': 2,
                    'Type': 'WORD'},
                   {'Confidence': 99.26438903808594,
                    'DetectedText': 'keep',
                    'Geometry': {'BoundingBox': {'Height': 0.0818721204996109,
                                                 'Left': 0.7344760298728943,
                                                 'Top': 0.5280686020851135,
                                                 'Width': 0.10748066753149033},
                                 'Polygon': [{'X': 0.7349520921707153,
                                              'Y': 0.5280686020851135},
                                             {'X': 0.8419566750526428,
                                              'Y': 0.5295097827911377},
                                             {'X': 0.8414806127548218,
                                              'Y': 0.6099407076835632},
                                             {'X': 0.7344760298728943,
                                              'Y': 0.6084995269775391}]},
                    'Id': 7,
                    'ParentId': 2,
                    'Type': 'WORD'},
                   {'Confidence': 88.95134735107422,
                    'DetectedText': 'Smiling',
                    'Geometry': {'BoundingBox': {'Height': 0.4326171875,
                                                 'Left': 0.46289217472076416,
                                                 'Top': 0.5634765625,
                                                 'Width': 0.5371078252792358},
                                 'Polygon': [{'X': 0.46289217472076416,
                                              'Y': 0.5634765625},
                                             {'X': 1.0, 'Y': 0.5634765625},
                                             {'X': 1.0, 'Y': 0.99609375},
                                             {'X': 0.46289217472076416,
                                              'Y': 0.99609375}]},
                    'Id': 8,
                    'ParentId': 3,
                    'Type': 'WORD'}],
 'TextModelVersion': '3.0'}
```

Detección de texto en un vídeo almacenado

La detección de texto de Amazon Rekognition Video en vídeos almacenados una operación asíncrona. Para empezar a detectar texto, llame StartTextDetection. Amazon Rekognition Video

publica el estado de finalización de una operación de análisis de vídeo en un tema de Amazon SNS. Si el análisis de vídeo es correcto, llame a GetTextDetection para obtener los resultados del análisis. Para obtener más información sobre cómo iniciar el análisis de vídeo y obtener los resultados, consulte Cómo llamar a las operaciones de Amazon Rekognition Video.

Este procedimiento se expande en el código que se describe en Análisis de un vídeo almacenado en un bucket de Amazon S3 con Java o Python (SDK). Utilice una cola de Amazon SQS para obtener el estado de finalización de una solicitud de análisis de vídeo.

Para detectar texto en un vídeo almacenado en un bucket de Amazon S3 (SDK)

1. Realice los pasos que se indican en Análisis de un vídeo almacenado en un bucket de Amazon S3 con Java o Python (SDK).
2. Agregue el siguiente código a la clase `VideoDetect` en el paso 1.

   Java

```
//Copyright 2018 Amazon.com, Inc. or its affiliates. All Rights Reserved.
//PDX-License-Identifier: MIT-0 (For details, see https://github.com/awsdocs/
amazon-rekognition-developer-guide/blob/master/LICENSE-SAMPLECODE.)


private static void StartTextDetection(String bucket, String video) throws
 Exception{

    NotificationChannel channel= new NotificationChannel()
            .withSNSTopicArn(snsTopicArn)
            .withRoleArn(roleArn);

    StartTextDetectionRequest req = new StartTextDetectionRequest()
            .withVideo(new Video()
                    .withS3Object(new S3Object()
                        .withBucket(bucket)
                        .withName(video)))
            .withNotificationChannel(channel);


    StartTextDetectionResult startTextDetectionResult =
 rek.startTextDetection(req);
    startJobId=startTextDetectionResult.getJobId();

}
```

```
private static void GetTextDetectionResults() throws Exception{

    int maxResults=10;
    String paginationToken=null;
    GetTextDetectionResult textDetectionResult=null;

    do{
        if (textDetectionResult !=null){
            paginationToken = textDetectionResult.getNextToken();

        }


        textDetectionResult = rek.getTextDetection(new
 GetTextDetectionRequest()
             .withJobId(startJobId)
             .withNextToken(paginationToken)
             .withMaxResults(maxResults));

        VideoMetadata videoMetaData=textDetectionResult.getVideoMetadata();

        System.out.println("Format: " + videoMetaData.getFormat());
        System.out.println("Codec: " + videoMetaData.getCodec());
        System.out.println("Duration: " + videoMetaData.getDurationMillis());
        System.out.println("FrameRate: " + videoMetaData.getFrameRate());


        //Show text, confidence values
        List<TextDetectionResult> textDetections =
 textDetectionResult.getTextDetections();


        for (TextDetectionResult text: textDetections) {
            long seconds=text.getTimestamp()/1000;
            System.out.println("Sec: " + Long.toString(seconds) + " ");
            TextDetection detectedText=text.getTextDetection();

            System.out.println("Text Detected: " +
 detectedText.getDetectedText());
                System.out.println("Confidence: " +
 detectedText.getConfidence().toString());
                System.out.println("Id : " + detectedText.getId());
                System.out.println("Parent Id: " + detectedText.getParentId());
```

```
                System.out.println("Bounding Box" +
 detectedText.getGeometry().getBoundingBox().toString());
                System.out.println("Type: " + detectedText.getType());
                System.out.println();
        }
    } while (textDetectionResult !=null && textDetectionResult.getNextToken() !=
 null);


}
```

En la función `main`, reemplace las líneas:

```
        StartLabelDetection(amzn-s3-demo-bucket, video);

        if (GetSQSMessageSuccess()==true)
         GetLabelDetectionResults();
```

por:

```
        StartTextDetection(amzn-s3-demo-bucket, video);

        if (GetSQSMessageSuccess()==true)
         GetTextDetectionResults();
```

Java V2

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

```
//snippet-start:[rekognition.java2.recognize_video_text.import]
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.S3Object;
import software.amazon.awssdk.services.rekognition.model.NotificationChannel;
import software.amazon.awssdk.services.rekognition.model.Video;
import
 software.amazon.awssdk.services.rekognition.model.StartTextDetectionRequest;
import
 software.amazon.awssdk.services.rekognition.model.StartTextDetectionResponse;
```

```
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import
 software.amazon.awssdk.services.rekognition.model.GetTextDetectionResponse;
import
 software.amazon.awssdk.services.rekognition.model.GetTextDetectionRequest;
import software.amazon.awssdk.services.rekognition.model.VideoMetadata;
import software.amazon.awssdk.services.rekognition.model.TextDetectionResult;
import java.util.List;
//snippet-end:[rekognition.java2.recognize_video_text.import]

/**
* Before running this Java V2 code example, set up your development environment,
 including your credentials.
*
* For more information, see the following documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
*/
public class DetectTextVideo {

 private static String startJobId ="";
 public static void main(String[] args) {

     final String usage = "\n" +
         "Usage: " +
         "   <bucket> <video> <topicArn> <roleArn>\n\n" +
         "Where:\n" +
         "   bucket - The name of the bucket in which the video is located (for
 example, (for example, amzn-s3-demo-bucket). \n\n"+
         "   video - The name of video (for example, people.mp4). \n\n" +
         "   topicArn - The ARN of the Amazon Simple Notification Service
 (Amazon SNS) topic. \n\n" +
         "   roleArn - The ARN of the AWS Identity and Access Management (IAM)
 role to use. \n\n" ;

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

     String bucket = args[0];
     String video = args[1];
     String topicArn = args[2];
```

```
    String roleArn = args[3];

    Region region = Region.US_EAST_1;
    RekognitionClient rekClient = RekognitionClient.builder()
        .region(region)
        .credentialsProvider(ProfileCredentialsProvider.create("profile-name"))
        .build();

    NotificationChannel channel = NotificationChannel.builder()
        .snsTopicArn(topicArn)
        .roleArn(roleArn)
        .build();

    startTextLabels(rekClient, channel, bucket, video);
    GetTextResults(rekClient);
    System.out.println("This example is done!");
    rekClient.close();
}

// snippet-start:[rekognition.java2.recognize_video_text.main]
public static void startTextLabels(RekognitionClient rekClient,
                                   NotificationChannel channel,
                                   String bucket,
                                   String video) {
    try {
        S3Object s3Obj = S3Object.builder()
            .bucket(bucket)
            .name(video)
            .build();

        Video vidOb = Video.builder()
            .s3Object(s3Obj)
            .build();

        StartTextDetectionRequest labelDetectionRequest =
StartTextDetectionRequest.builder()
            .jobTag("DetectingLabels")
            .notificationChannel(channel)
            .video(vidOb)
            .build();

        StartTextDetectionResponse labelDetectionResponse =
rekClient.startTextDetection(labelDetectionRequest);
        startJobId = labelDetectionResponse.jobId();
```

```
    } catch (RekognitionException e) {
        System.out.println(e.getMessage());
        System.exit(1);
    }
}

public static void GetTextResults(RekognitionClient rekClient) {

    try {
        String paginationToken=null;
        GetTextDetectionResponse textDetectionResponse=null;
        boolean finished = false;
        String status;
        int yy=0 ;

        do{
            if (textDetectionResponse !=null)
                paginationToken = textDetectionResponse.nextToken();

            GetTextDetectionRequest recognitionRequest =
GetTextDetectionRequest.builder()
                .jobId(startJobId)
                .nextToken(paginationToken)
                .maxResults(10)
                .build();

            // Wait until the job succeeds.
            while (!finished) {
                textDetectionResponse =
rekClient.getTextDetection(recognitionRequest);
                status = textDetectionResponse.jobStatusAsString();

                if (status.compareTo("SUCCEEDED") == 0)
                    finished = true;
                else {
                    System.out.println(yy + " status is: " + status);
                    Thread.sleep(1000);
                }
                yy++;
            }

            finished = false;
```

```
            // Proceed when the job is done - otherwise VideoMetadata is null.
            VideoMetadata videoMetaData=textDetectionResponse.videoMetadata();
            System.out.println("Format: " + videoMetaData.format());
            System.out.println("Codec: " + videoMetaData.codec());
            System.out.println("Duration: " + videoMetaData.durationMillis());
            System.out.println("FrameRate: " + videoMetaData.frameRate());
            System.out.println("Job");

            List<TextDetectionResult> labels=
 textDetectionResponse.textDetections();
            for (TextDetectionResult detectedText: labels) {
                System.out.println("Confidence: " +
 detectedText.textDetection().confidence().toString());
                System.out.println("Id : " +
 detectedText.textDetection().id());
                System.out.println("Parent Id: " +
 detectedText.textDetection().parentId());
                System.out.println("Type: " +
 detectedText.textDetection().type());
                System.out.println("Text: " +
 detectedText.textDetection().detectedText());
                System.out.println();
            }

        } while (textDetectionResponse !=null &&
 textDetectionResponse.nextToken() != null);

    } catch(RekognitionException | InterruptedException e) {
        System.out.println(e.getMessage());
        System.exit(1);
    }
 }
 // snippet-end:[rekognition.java2.recognize_video_text.main]
}
```

Python

```
#Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
#PDX-License-Identifier: MIT-0 (For details, see https://github.com/awsdocs/
amazon-rekognition-developer-guide/blob/master/LICENSE-SAMPLECODE.)

    def StartTextDetection(self):
```

```
        response=self.rek.start_text_detection(Video={'S3Object': {'Bucket':
 self.bucket, 'Name': self.video}},
            NotificationChannel={'RoleArn': self.roleArn, 'SNSTopicArn':
 self.snsTopicArn})

        self.startJobId=response['JobId']
        print('Start Job Id: ' + self.startJobId)

    def GetTextDetectionResults(self):
        maxResults = 10
        paginationToken = ''
        finished = False

        while finished == False:
            response = self.rek.get_text_detection(JobId=self.startJobId,
                                        MaxResults=maxResults,
                                        NextToken=paginationToken)

            print('Codec: ' + response['VideoMetadata']['Codec'])

            print('Duration: ' + str(response['VideoMetadata']
['DurationMillis']))
            print('Format: ' + response['VideoMetadata']['Format'])
            print('Frame rate: ' + str(response['VideoMetadata']['FrameRate']))
            print()

            for textDetection in response['TextDetections']:
                text=textDetection['TextDetection']

                print("Timestamp: " + str(textDetection['Timestamp']))
                print("   Text Detected: " + text['DetectedText'])
                print("   Confidence: " +  str(text['Confidence']))
                print ("      Bounding box")
                print ("        Top: " + str(text['Geometry']['BoundingBox']
['Top']))
                print ("        Left: " + str(text['Geometry']['BoundingBox']
['Left']))
                print ("        Width: " +  str(text['Geometry']['BoundingBox']
['Width']))
                print ("        Height: " +  str(text['Geometry']['BoundingBox']
['Height']))
                print ("   Type: " + str(text['Type']) )
                print()
```

```
            if 'NextToken' in response:
                paginationToken = response['NextToken']
            else:
                finished = True
```

En la función `main`, reemplace las líneas:

```
    analyzer.StartLabelDetection()
    if analyzer.GetSQSMessageSuccess()==True:
        analyzer.GetLabelDetectionResults()
```

por:

```
    analyzer.StartTextDetection()
    if analyzer.GetSQSMessageSuccess()==True:
        analyzer.GetTextDetectionResults()
```

CLI

Ejecute el siguiente AWS CLI comando para empezar a detectar texto en un vídeo.

```
 aws rekognition start-text-detection --video "{"S3Object":{"Bucket":"amzn-s3-
demo-bucket","Name":"video-name"}}"\
 --notification-channel "{"SNSTopicArn":"topic-arn","RoleArn":"role-arn"}" \
 --region region-name --profile profile-name
```

Actualice los siguientes valores:

- Cambie `amzn-s3-demo-bucket` y `video-name` por el nombre del bucket de Amazon S3 y el nombre de archivo que especificó en el paso 2.
- Cambie `region-name` por la región de AWS que está utilizando.
- Sustituya el valor de `profile-name` de por el nombre de su perfil de desarrollador.
- Reemplace `topic-ARN` por el ARN del tema de Amazon SNS que creó en el paso 3 de Configuración de Amazon Rekognition Video.
- Cambie `role-ARN` por el ARN del rol de servicio de IAM que creó en el paso 7 de Configuración de Amazon Rekognition Video.

Si accede a la CLI en un dispositivo Windows, utilice comillas dobles en lugar de comillas simples y evite las comillas dobles internas con una barra invertida (es decir, \) para corregir cualquier error del analizador que pueda encontrar. Consulte a continuación un ejemplo:

```
aws rekognition start-text-detection --video \
 "{\"S3Object\":{\"Bucket\":\"amzn-s3-demo-bucket\",\"Name\":\"video-name\"}}" \
 --notification-channel "{\"SNSTopicArn\":\"topic-arn\",\"RoleArn\":\"role-arn
\"}" \
 --region region-name --profile profile-name
```

Tras ejecutar el ejemplo de código de procedimiento, copie el `jobID` devuelto y envíelo al siguiente comando `GetTextDetection` para obtener los resultados, sustituyendo `job-id-number` por el `jobID` que recibió anteriormente:

```
aws rekognition get-text-detection --job-id job-id-number --profile profile-name
```

> **Note**
>
> Si ya ha ejecutado un ejemplo de vídeo distinto de Análisis de un vídeo almacenado en un bucket de Amazon S3 con Java o Python (SDK), el código que se va a reemplazar podría ser diferente.

3. Ejecute el código. El texto que se detectó en el vídeo se muestra en una lista.

Filtros

Los filtros son parámetros de solicitud opcionales que se pueden usar cuando se llama a `StartTextDetection`. El filtrado por región, tamaño y puntuación de confianza del texto le brinda una flexibilidad adicional para controlar el resultado de su detección del texto. Gracias a las regiones de interés, puede limitar la detección de texto a las regiones que considera relevantes: por ejemplo, el tercio inferior para gráficos o la esquina superior izquierda para mostrar marcadores en un partido de fútbol. El filtro de tamaño del cuadro delimitador de texto puede utilizarse para evitar texto en segundo plano pequeño que sea molesto o irrelevante. Por último, el filtro de confianza del texto le permite eliminar resultados que no sean fiables, como texto borroso o difuminado.

Para obtener información sobre los valores de filtro, consulte `DetectTextFilters`.

Puede utilizar los siguientes filtros:

- MinConfidence—Establece el nivel de confianza de la detección de palabras. Las palabras con confianza de detección por debajo de este nivel se excluyen del resultado. Los valores deben estar comprendidos entre 0 y 100.
- MinBoundingBoxWidth— Establece el ancho mínimo del cuadro delimitador de palabras. Las palabras con cuadros delimitadores menores que este valor se excluyen del resultado. El valor es relativo al ancho del fotograma de vídeo.
- MinBoundingBoxHeight— Establece la altura mínima del cuadro delimitador de palabras. Las palabras con alturas de cuadro delimitador inferiores a este valor se excluyen del resultado. El valor es relativo a la altura del fotograma de vídeo.
- RegionsOfInterest— Limita la detección a una región específica del marco. Los valores son relativos a las dimensiones del fotograma. Para objetos que se encuentra solo parcialmente dentro de las regiones, la respuesta es indefinida.

GetTextDetection respuesta

`GetTextDetection` devuelve una matriz (`TextDetectionResults`) que contiene información sobre el texto detectado en el vídeo. Un elemento de la matriz, TextDetection, existe para cada vez que se detecta una palabra o línea en el vídeo. Los elementos de la matriz se devuelven ordenados por tiempo (en milisegundos) desde el comienzo del vídeo.

El siguiente ejemplo es una respuesta JSON parcial de `GetTextDetection`. En la respuesta, tenga en cuenta lo siguiente:

- Información de texto: el elemento de la `TextDetectionResult` matriz contiene información sobre el texto detectado (TextDetection) y la hora en que se detectó el texto en el vídeo (`Timestamp`).
- Información de paginación: el ejemplo muestra una página de información de detección de texto. Puede especificar la cantidad de elementos de texto que se van a devolver en el parámetro de entrada `MaxResults` para `GetTextDetection`. Si hay más resultados que el valor de `MaxResults` o hay más resultados que el máximo predeterminado, `GetTextDetection` devuelve un token (`NextToken`) que se utiliza para obtener la siguiente página de resultados. Para obtener más información, consulte Obtención de los resultados del análisis de Amazon Rekognition Video.

- Información de vídeo: la respuesta incluye información acerca del formato de vídeo (`VideoMetadata`) de cada página de información devuelta por `GetTextDetection`.


```
{
    "JobStatus": "SUCCEEDED",
    "VideoMetadata": {
        "Codec": "h264",
        "DurationMillis": 174441,
        "Format": "QuickTime / MOV",
        "FrameRate": 29.970029830932617,
        "FrameHeight": 480,
        "FrameWidth": 854
    },
    "TextDetections": [
        {
            "Timestamp": 967,
            "TextDetection": {
                "DetectedText": "Twinkle Twinkle Little Star",
                "Type": "LINE",
                "Id": 0,
                "Confidence": 99.91780090332031,
                "Geometry": {
                    "BoundingBox": {
                        "Width": 0.8337579369544983,
                        "Height": 0.08365312218666077,
                        "Left": 0.08313830941915512,
                        "Top": 0.4663468301296234
                    },
                    "Polygon": [
                        {
                            "X": 0.08313830941915512,
                            "Y": 0.4663468301296234
                        },
                        {
                            "X": 0.9168962240219116,
                            "Y": 0.4674469828605652
                        },
                        {
                            "X": 0.916861355304718,
                            "Y": 0.5511001348495483
                        },
```

```
                    {
                        "X": 0.08310343325138092,
                        "Y": 0.5499999523162842
                    }
                ]
            }
        }
    },
    {
        "Timestamp": 967,
        "TextDetection": {
            "DetectedText": "Twinkle",
            "Type": "WORD",
            "Id": 1,
            "ParentId": 0,
            "Confidence": 99.98338317871094,
            "Geometry": {
                "BoundingBox": {
                    "Width": 0.2423887550830841,
                    "Height": 0.0833333358168602,
                    "Left": 0.08313817530870438,
                    "Top": 0.46666666865348816
                },
                "Polygon": [
                    {
                        "X": 0.08313817530870438,
                        "Y": 0.46666666865348816
                    },
                    {
                        "X": 0.3255269229412079,
                        "Y": 0.46666666865348816
                    },
                    {
                        "X": 0.3255269229412079,
                        "Y": 0.550000011920929
                    },
                    {
                        "X": 0.08313817530870438,
                        "Y": 0.550000011920929
                    }
                ]
            }
        }
    },
```

```
        {
            "Timestamp": 967,
            "TextDetection": {
                "DetectedText": "Twinkle",
                "Type": "WORD",
                "Id": 2,
                "ParentId": 0,
                "Confidence": 99.982666015625,
                "Geometry": {
                    "BoundingBox": {
                        "Width": 0.2423887550830841,
                        "Height": 0.08124999701976776,
                        "Left": 0.3454332649707794,
                        "Top": 0.46875
                    },
                    "Polygon": [
                        {
                            "X": 0.3454332649707794,
                            "Y": 0.46875
                        },
                        {
                            "X": 0.5878220200538635,
                            "Y": 0.46875
                        },
                        {
                            "X": 0.5878220200538635,
                            "Y": 0.550000011920929
                        },
                        {
                            "X": 0.3454332649707794,
                            "Y": 0.550000011920929
                        }
                    ]
                }
            }
        },
        {
            "Timestamp": 967,
            "TextDetection": {
                "DetectedText": "Little",
                "Type": "WORD",
                "Id": 3,
                "ParentId": 0,
                "Confidence": 99.8787612915039,
```

```
                "Geometry": {
                    "BoundingBox": {
                        "Width": 0.16627635061740875,
                        "Height": 0.08124999701976776,
                        "Left": 0.6053864359855652,
                        "Top": 0.46875
                    },
                    "Polygon": [
                        {
                            "X": 0.6053864359855652,
                            "Y": 0.46875
                        },
                        {
                            "X": 0.7716627717018127,
                            "Y": 0.46875
                        },
                        {
                            "X": 0.7716627717018127,
                            "Y": 0.550000011920929
                        },
                        {
                            "X": 0.6053864359855652,
                            "Y": 0.550000011920929
                        }
                    ]
                }
            }
        },
        {
            "Timestamp": 967,
            "TextDetection": {
                "DetectedText": "Star",
                "Type": "WORD",
                "Id": 4,
                "ParentId": 0,
                "Confidence": 99.82640075683594,
                "Geometry": {
                    "BoundingBox": {
                        "Width": 0.12997658550739288,
                        "Height": 0.08124999701976776,
                        "Left": 0.7868852615356445,
                        "Top": 0.46875
                    },
                    "Polygon": [
```

```
                        {
                            "X": 0.7868852615356445,
                            "Y": 0.46875
                        },
                        {
                            "X": 0.9168618321418762,
                            "Y": 0.46875
                        },
                        {
                            "X": 0.9168618321418762,
                            "Y": 0.550000011920929
                        },
                        {
                            "X": 0.7868852615356445,
                            "Y": 0.550000011920929
                        }
                    ]
                }
            }
        }
    ],
    "NextToken": "NiHpGbZFnkM/S8kLcukMni15wb05iKtquu/Mwc+Qg1LVlMjjKNOD0Z0GusSPg7TONLe
+OZ3P",
    "TextModelVersion": "3.0"
}
```

Detección de segmentos de vídeo en vídeo almacenado

Amazon Rekognition Video proporciona una API que identifica segmentos de vídeo útiles, como fotogramas negros y créditos finales.

Los espectadores están viendo más contenidos que nunca. En particular, las plataformas Over-The-Top (OTT) y Video-On-Demand (VOD) ofrecen una amplia selección de opciones de contenido en cualquier momento, en cualquier lugar y en cualquier pantalla. Con la proliferación de volúmenes de contenido, las empresas de medios se enfrentan a desafíos en la preparación y administración de su contenido. Esto es fundamental para proporcionar una experiencia de visualización de alta calidad y una mejor monetización del contenido. Hoy en día, las empresas utilizan grandes equipos de mano de obra humana capacitada para realizar tareas como las siguientes.

- Buscar dónde comienzan los créditos iniciales y finales en un fragmento de contenido
- Elegir los lugares correctos para insertar anuncios, por ejemplo, en secuencias silenciosas de cuadros negros
- Descomponer los vídeos en clips más pequeños para mejorar la indexación

Estos procesos manuales son costosos, lentos y no se pueden escalar para mantenerse al día con el volumen de contenido producido, con licencia y recuperado diariamente de los archivos.

Puede utilizar Amazon Rekognition Video para automatizar las tareas operativas de análisis multimedia mediante una APIs detección de segmentos de vídeo totalmente gestionada y diseñada específicamente con tecnología de aprendizaje automático (ML). Al utilizar el APIs segmento Amazon Rekognition Video, puede analizar fácilmente grandes volúmenes de vídeos y detectar marcadores como fotogramas negros o cambios de toma. Obtendrá códigos de tiempo, marcas temporales y números de fotogramas de SMPTE (Society of Motion Picture and Television Engineers) para cada detección. No se requiere experiencia en machine learning.

Amazon Rekognition Video analiza vídeos almacenados en un bucket de Amazon Simple Storage Service (Amazon S3). Los códigos de tiempo de SMPTE que se devuelven son precisos a nivel de fotograma: Amazon Rekognition Video proporciona el número exacto de fotogramas de un segmento de vídeo detectado y gestiona varios formatos de velocidad de fotogramas de vídeo de forma automática. Puede utilizar los metadatos con precisión hasta el fotograma de Amazon Rekognition Video, para automatizar ciertas tareas por completo o reducir significativamente la carga de trabajo de revisión de operadores humanos capacitados, de modo que puedan centrarse en un trabajo más

creativo. Puede realizar tareas como preparar contenido, insertar anuncios y agregar "marcadores" al contenido a gran escala en la nube.

Para obtener información sobre los precios, consulte Precios de Amazon Rekognition.

La detección de segmentos de Amazon Rekognition Video admite dos tipos de tareas de segmentación: detección de Tomas técnicas y Detección de tomas.

Temas

- Tomas técnicas
- Detección de tomas
- Acerca de la API de detección de segmentos de Amazon Rekognition Video
- Uso de la API de segmentos de Amazon Rekognition
- Ejemplo: Detección de segmentos en un vídeo almacenado

Tomas técnicas

Una indicación técnica identifica los fotogramas negros, las barras de color, los créditos iniciales, los créditos finales, los logotipos de los estudios y el contenido principal del programa en un vídeo.

Fotogramas negros

Los vídeos suelen contener fotogramas negros, vacíos y sin audio que se utilizan como indicaciones para insertar anuncios o a fin de delimitar el final de un segmento de programa, como una escena o los créditos de apertura. Con Amazon Rekognition Video, se pueden detectar tales secuencias de fotogramas negros para automatizar la inserción de anuncios, empaquetar el contenido para vídeo bajo demanda y delimitar varios segmentos o escenas de programas. Los fotogramas negros con audio (como los fundidos o las voces en off) se consideran como contenido y no se devuelven.

Créditos

Amazon Rekognition Video puede ayudarle a identificar de forma automática los fotogramas exactos en los que comienzan y terminan los créditos iniciales y finales de una película o un programa de televisión. Con esta información, puede generar «marcadores de maratones» o mensajes interactivos para el espectador, como «Próximo episodio» u «Omitir introducción», en aplicaciones de vídeo en diferido (VOD). También puede detectar el primer y el último fotograma del contenido del programa

en un vídeo. Amazon Rekognition Video está capacitado para gestionar una amplia variedad de estilos de créditos iniciales y finales, que van desde simples créditos continuos hasta créditos más desafiantes junto con el contenido.

Barras de color

Amazon Rekognition Video permite detectar secciones de vídeo que muestran barras de color según la SMPTE, que son un conjunto de colores mostrados en patrones específicos para garantizar que el color está calibrado correctamente en los monitores de transmisión, programas y en las cámaras. Para obtener más información acerca de las barras de color SMPTE, consulte Barra de color SMPTE. Estos metadatos sirven para preparar el contenido para las aplicaciones de vídeo bajo demanda mediante la eliminación de segmentos de barras de color del contenido o para detectar problemas como la pérdida de señales de emisión en una grabación, cuando las barras de color se muestran continuamente como una señal predeterminada en lugar de contenido.

Caretas

Las caretas son secciones del vídeo, normalmente cerca del principio, que contienen metadatos de texto sobre el episodio, el estudio, el formato de vídeo, los canales de audio y mucho más. Amazon Rekognition Video puede identificar el inicio y el final de las caretas, lo que facilita el uso de los metadatos de texto o la eliminación de la careta al preparar el contenido para su visualización final.

Logotipos de estudio

Los logotipos de estudio son secuencias que muestran los logotipos o emblemas del estudio de producción que participó en la realización del espectáculo. Amazon Rekognition Video puede detectar estas secuencias para que los usuarios puedan revisarlas e identificar los estudios.

Contenidos

El contenido son las partes del programa de televisión o película que contienen el programa o elementos relacionados. Los fotogramas negros, los créditos, las barras de colores, las caretas y los logotipos de los estudios no se consideran contenido. Amazon Rekognition Video puede detectar el inicio y el final de cada segmento de contenido del vídeo, por lo que puede encontrar el tiempo de ejecución del programa o segmentos específicos.

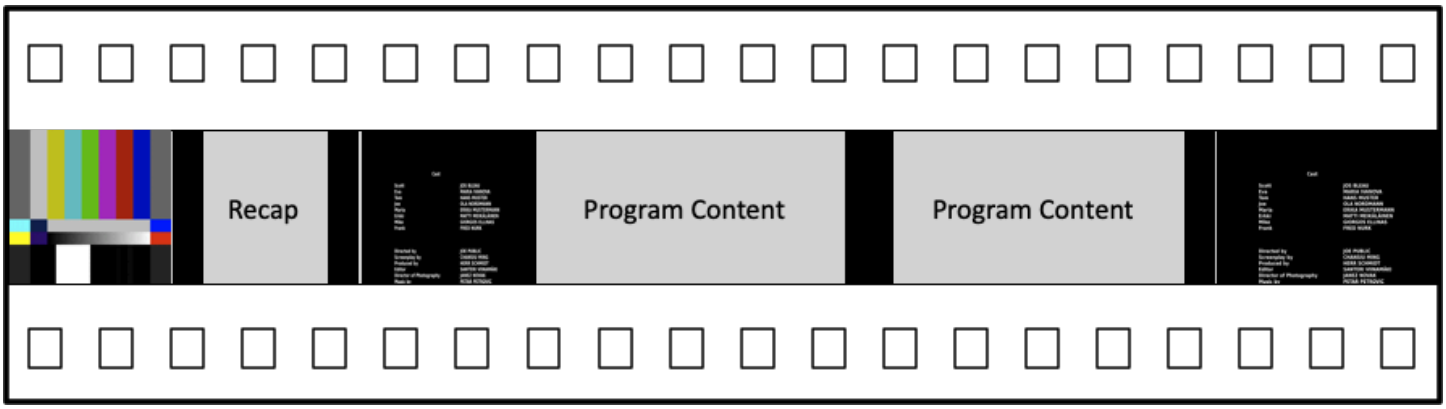
El contenido incluye, entre otros, lo siguiente:

- Escenas del programa entre dos pausas publicitarias

- Un resumen rápido del episodio anterior al principio del vídeo
- Contenido adicional posterior a los créditos
- Contenido «sin texto», como un conjunto de todas las escenas del programa que originalmente contenían texto superpuesto, pero en el que se ha eliminado el texto para poder traducirlo a otros idiomas.

Una vez que Amazon Rekognition Video termine de detectar todos los segmentos de contenido, puede aplicar los conocimientos del dominio o enviarlos para que los revisen un humano a fin de categorizar cada segmento con más detalle. Por ejemplo, si utiliza vídeos que siempre comienzan con un resumen, puede clasificar el primer segmento de contenido como un resumen.

En el siguiente diagrama se muestran los segmentos técnicos de referencia en la escala de tiempo de una serie o película. Tenga en cuenta las barras de colores y los créditos iniciales, los segmentos de contenido, como el resumen y el programa principal, los fotogramas negros que aparecen en todo el vídeo y los créditos finales.



Detección de tomas

Una toma es una serie de imágenes consecutivas interrelacionadas que se capturan contiguamente por una sola cámara y que representan una acción continua en el tiempo y el espacio. Con Amazon Rekognition Video, se puede detectar el inicio, el final y la duración de cada toma, así como contabilizar todas las tomas de un contenido. Puede utilizar metadatos de toma para tareas como las siguientes.

- Creación de vídeos promocionales utilizando tomas seleccionadas.
- Inserción de anuncios en lugares que no interrumpan la experiencia del espectador, como en medio de una toma cuando alguien habla.

- Generar un conjunto de miniaturas de vista previa que impidan el contenido de transición entre tomas.

Una detección de toma se marca en el fotograma exacto donde hay un cambio brusco a otra cámara. Si hay una transición suave de una cámara a otra, Amazon Rekognition Video omite la transición. Esto garantiza que los tiempos de inicio y finalización de toma no incluyan secciones sin contenido real.

En el siguiente diagrama se ilustran los segmentos de detección de tomas en una tira de película. Tenga en cuenta que cada toma se identifica mediante un corte desde un ángulo o ubicación de cámara al siguiente.



Acerca de la API de detección de segmentos de Amazon Rekognition Video

Para segmentar un vídeo almacenado, utilice las operaciones asíncronas StartSegmentDetectiony de GetSegmentDetectionAPI para iniciar un trabajo de segmentación y obtener los resultados. La detección de segmentos acepta vídeos almacenados en un bucket de Amazon S3 y devuelve una salida JSON. Puede elegir detectar solo indicaciones técnicas, solo cambios de toma o ambas opciones si configura la solicitud de la API `StartSegmentdetection`. También puede filtrar los segmentos detectados si establece umbrales para una confianza de predicción mínima. Para obtener más información, consulte Uso de la API de segmentos de Amazon Rekognition. Para ver el código de ejemplo, consulte Ejemplo: Detección de segmentos en un vídeo almacenado.

Uso de la API de segmentos de Amazon Rekognition

La detección de segmentos de Amazon Rekognition Video en vídeos almacenados es una operación asíncrona de Amazon Rekognition Video. La API de Amazon Rekognition Segment es una API

compuesta en la que se elige el tipo de análisis (indicaciones técnicas o detección de tomas) de una sola llamada a la API. Para obtener información acerca de cómo llamar a operaciones asíncronas, consulte Cómo llamar a las operaciones de Amazon Rekognition Video.

Temas

- Inicio del análisis de segmentos
- Obtención de los resultados del análisis de segmentos

Inicio del análisis de segmentos

Para iniciar la detección de segmentos en una videollamada almacenada StartSegmentDetection. Los parámetros de entrada son los mismos que otras operaciones Amazon Rekognition Video con la adición de la selección de tipo de segmento y filtrado de resultados. Para obtener más información, consulte Comenzar el análisis de vídeo.

El siguiente es el ejemplo JSON que pasa `StartSegmentDetection`. La solicitud especifica que se detecten tanto los segmentos de detección de indicaciones técnicas como los de detección de tomas. Se solicitan diferentes filtros para la confianza mínima de detección de los segmentos de indicaciones técnicas (90 %) y segmentos de detección de tomas (80 %).

```
{
  "Video": {
    "S3Object": {
      "Bucket": "test_files",
      "Name": "test_file.mp4"
    }
    "SegmentTypes":["TECHNICAL_CUES", "SHOT"]
    "Filters": {
      "TechnicalCueFilter": {
         "MinSegmentConfidence": 90,
         "BlackFrame" : {
            "MaxPixelThreshold": 0.1,
            "MinCoveragePercentage": 95
         }
      },
      "ShotFilter" : {
          "MinSegmentConfidence": 60
      }
  }
}
```

Selección de un tipo de segmento

Utilice el parámetro de entrada de `SegmentTypes` matriz para detectar segmentos de detección de and/or tomas de referencia técnicas en el vídeo de entrada.

- TECHNICAL_CUE: identifica las marcas de tiempo con precisión a nivel de fotograma para el inicio, el final y la duración de las señales técnicas (fotogramas negros, barras de colores, créditos iniciales, créditos finales, logotipos de estudio y contenido principal del programa) detectadas en un vídeo. Por ejemplo, puede utilizar indicaciones técnicas para encontrar el inicio de los créditos finales. Para obtener más información, consulte Tomas técnicas.
- TOMA: identifica el inicio, el final y la duración de una toma. Por ejemplo, puede utilizar la detección de tomas para identificar las tomas candidatas para la edición final de un vídeo. Para obtener más información, consulte Detección de tomas.

Filtrado de los resultados del análisis

Puede utilizar el parámetro de entrada `Filters` (StartSegmentDetectionFilters) para especificar la confianza de detección mínima que se devuelve en la respuesta. En `Filters`, use `ShotFilter` (StartShotDetectionFilter) para filtrar las tomas detectadas. Utilice `TechnicalCueFilter` (StartTechnicalCueDetectionFilter) para filtrar las señales técnicas.

Para ver el código de ejemplo, consulte Ejemplo: Detección de segmentos en un vídeo almacenado.

Obtención de los resultados del análisis de segmentos

Amazon Rekognition Video publica el estado de finalización de una operación de análisis de vídeo en un tema de Amazon Simple Notification Service. Si el análisis de vídeo se realiza correctamente, llame GetSegmentDetectionpara obtener los resultados del análisis de vídeo.

A continuación, se muestra un ejemplo de solicitud `GetSegmentDetection`. `JobId` es el identificador de trabajo que devuelve la llamada a `StartSegmentDetection`. Para obtener información acerca del resto de parámetros de entrada, consulte Obtención de los resultados del análisis de Amazon Rekognition Video.

```
{
    "JobId": "270c1cc5e1d0ea2fbc59d97cb69a72a5495da75851976b14a1784ca90fc180e3",
    "MaxResults": 10,
    "NextToken": "XfXnZKiyMOGDhzBzYUhS5puM+g1IgezqFeYpv/H/+5noP/LmM57FitUAwSQ5D6G4AB/
PNwolrw=="
```

```
}
```

`GetSegmentDetection` devuelve los resultados del análisis solicitado e información general sobre el vídeo almacenado.

Información general

`GetSegmentDection` devuelve la siguiente información general.

- Información de audio: la respuesta incluye metadatos de audio en una matriz de AudioMetadataobjetos. `AudioMetadata` Puede haber varias secuencias de audio. Cada objeto `AudioMetadata` contiene metadatos para una sola secuencia de audio. La información de audio de un objeto `AudioMetadata` incluye el códec de audio, el número de canales de audio, la duración de la transmisión de audio y la frecuencia de muestreo. Los metadatos de audio se devuelven en cada página de información devuelta por `GetSegmentDetection`.
- Información de vídeo: actualmente, Amazon Rekognition Video devuelve un VideoMetadataúnico objeto de la matriz. `VideoMetadata` El objeto contiene información sobre la secuencia de vídeo en el archivo de entrada que Amazon Rekognition Video ha elegido para analizar. El objeto `VideoMetadata` incluye el códec de vídeo, el formato de vídeo y otra información. Los metadatos de vídeo se devuelven en cada página de información devuelta por `GetSegmentDetection`.
- Información de paginación: el ejemplo muestra una página de información de segmentación. Puede especificar la cantidad de elementos que se van a devolver en el parámetro de entrada `MaxResults` para `GetSegmentDetection`. Si existen más resultados que `MaxResults`, `GetSegmentDetection` devuelve un token (`NextToken`) que se utiliza para obtener la siguiente página de resultados. Para obtener más información, consulte Obtención de los resultados del análisis de Amazon Rekognition Video.
- Solicitar información: el tipo de análisis solicitado en la llamada a `StartSegmentDetection` se devuelve en el campo `SelectedSegmentTypes`.

Segmentos

La información de indicaciones técnicas y de tomas detectadas en un vídeo se devuelven en una matriz, `Segments`, de objetos SegmentDetection. La matriz se ordena por los tipos de segmento (TECHNICAL_CUE o SHOT) especificados en el parámetro de entrada `SegmentTypes` de `StartSegmentDetection`. En cada tipo de segmento, la matriz se ordena por valores de marca temporal. Cada objeto `SegmentDetection` incluye información sobre el tipo de segmento detectado

(indicación técnica o detección de toma) e información general, como el tiempo de inicio, el tiempo de finalización y la duración del segmento.

La información de tiempo se devuelve en tres formatos.

- Milisegundos

  El número de milisegundos desde el inicio del vídeo. Los campos `DurationMillis`, `StartTimestampMillis` y `EndTimestampMillis` están en formato de milisegundos.
- Código de tiempo

  Los códigos temporales de Amazon Rekognition Video están en formato SMPTE donde cada fotograma de vídeo tiene un valor de código temporal único. El formato es hh:mm:ss:fotograma. Por ejemplo, un valor de código temporal de 01:05:40:07 se leería como una hora, cinco minutos, cuarenta segundos y siete fotogramas. Amazon Rekognition Video admite casos de uso de reducción de la velocidad de fotogramas. El formato de código temporal de velocidad de pérdida es hh:mm:ss;fotograma. Los campos `DurationSMPTE`, `StartTimecodeSMPTE` y `EndTimecodeSMPTE` están en formato de código temporal.
- Contadores de fotogramas

  La duración de cada segmento de vídeo también se expresa con el número de fotogramas. El campo `StartFrameNumber` proporciona el número de fotograma al principio de un segmento de vídeo y `EndFrameNumber` el número de fotograma al final de un segmento de vídeo. `DurationFrames` indica el número total de fotogramas de un segmento de vídeo. Estos valores se calculan mediante un índice de fotogramas que comienza por 0.

Puede utilizar el campo `SegmentType` para determinar el tipo de segmento que devuelve Amazon Rekognition Video.

- Indicaciones técnicas: el **`TechnicalCueSegment`** campo es un TechnicalCueSegmentobjeto que contiene la fiabilidad de la detección y el tipo de señal técnica. Los tipos de señales técnicas son `ColorBars`, `EndCredits`, `BlackFrames`, `OpeningCredits`, `StudioLogo`, `Slate` y `Content`.
- Captura: el `ShotSegment` campo es un ShotSegmentobjeto que contiene la confianza de detección y un identificador del segmento de la toma dentro del vídeo.

El siguiente ejemplo es la respuesta JSON de `GetSegmentDetection`.

```
{
```

```
    "SelectedSegmentTypes": [
        {
            "ModelVersion": "2.0",
            "Type": "SHOT"
        },
        {
            "ModelVersion": "2.0",
            "Type": "TECHNICAL_CUE"
        }
    ],
    "Segments": [
        {
            "DurationFrames": 299,
            "DurationSMPTE": "00:00:09;29",
            "StartFrameNumber": 0,
            "EndFrameNumber": 299,
            "EndTimecodeSMPTE": "00:00:09;29",
            "EndTimestampMillis": 9976,
            "StartTimestampMillis": 0,
            "DurationMillis": 9976,
            "StartTimecodeSMPTE": "00:00:00;00",
            "Type": "TECHNICAL_CUE",
            "TechnicalCueSegment": {
                "Confidence": 90.45006561279297,
                "Type": "BlackFrames"
            }
        },
        {
            "DurationFrames": 150,
            "DurationSMPTE": "00:00:05;00",
            "StartFrameNumber": 299,
            "EndFrameNumber": 449,
            "EndTimecodeSMPTE": "00:00:14;29",
            "EndTimestampMillis": 14981,
            "StartTimestampMillis": 9976,
            "DurationMillis": 5005,
            "StartTimecodeSMPTE": "00:00:09;29",
            "Type": "TECHNICAL_CUE",
            "TechnicalCueSegment": {
                "Confidence": 100.0,
                "Type": "Content"
            }
        },
        {
```

```
            "DurationFrames": 299,
            "ShotSegment": {
                "Index": 0,
                "Confidence": 99.9982681274414
            },
            "DurationSMPTE": "00:00:09;29",
            "StartFrameNumber": 0,
            "EndFrameNumber": 299,
            "EndTimecodeSMPTE": "00:00:09;29",
            "EndTimestampMillis": 9976,
            "StartTimestampMillis": 0,
            "DurationMillis": 9976,
            "StartTimecodeSMPTE": "00:00:00;00",
            "Type": "SHOT"
        },
        {
            "DurationFrames": 149,
            "ShotSegment": {
                "Index": 1,
                "Confidence": 99.9982681274414
            },
            "DurationSMPTE": "00:00:04;29",
            "StartFrameNumber": 300,
            "EndFrameNumber": 449,
            "EndTimecodeSMPTE": "00:00:14;29",
            "EndTimestampMillis": 14981,
            "StartTimestampMillis": 10010,
            "DurationMillis": 4971,
            "StartTimecodeSMPTE": "00:00:10;00",
            "Type": "SHOT"
        }
    ],
    "JobStatus": "SUCCEEDED",
    "VideoMetadata": [
        {
            "Format": "QuickTime / MOV",
            "FrameRate": 29.970029830932617,
            "Codec": "h264",
            "DurationMillis": 15015,
            "FrameHeight": 1080,
            "FrameWidth": 1920,
            "ColorRange": "LIMITED"

        }
```

```
    ],
    "AudioMetadata": [
        {
            "NumberOfChannels": 1,
            "SampleRate": 48000,
            "Codec": "aac",
            "DurationMillis": 15007
        }
    ]
}
```

Para ver el código de ejemplo, consulte Ejemplo: Detección de segmentos en un vídeo almacenado.

Ejemplo: Detección de segmentos en un vídeo almacenado

El siguiente procedimiento muestra cómo detectar segmentos de indicaciones técnicas y segmentos de detección de tomas en un vídeo almacenado en un bucket de Amazon S3. El procedimiento también muestra cómo filtrar los segmentos detectados en función de la confianza que Amazon Rekognition Video tiene en la precisión de la detección.

El ejemplo amplía el código en Análisis de un vídeo almacenado en un bucket de Amazon S3 con Java o Python (SDK) que utiliza una cola de Amazon Simple Queue Service para obtener el estado de realización de una solicitud de análisis de vídeo.

Para detectar segmentos en un vídeo almacenado en un bucket de Amazon S3 (SDK)

1. Realice Análisis de un vídeo almacenado en un bucket de Amazon S3 con Java o Python (SDK).
2. Agregue lo siguiente al código que ha utilizado en el paso 1.

   Java

   1. Agregue las siguientes importaciones:

```
import com.amazonaws.services.rekognition.model.GetSegmentDetectionRequest;
import com.amazonaws.services.rekognition.model.GetSegmentDetectionResult;
import com.amazonaws.services.rekognition.model.SegmentDetection;
import com.amazonaws.services.rekognition.model.SegmentType;
import com.amazonaws.services.rekognition.model.SegmentTypeInfo;
import com.amazonaws.services.rekognition.model.ShotSegment;
import
 com.amazonaws.services.rekognition.model.StartSegmentDetectionFilters;
```

```
import
 com.amazonaws.services.rekognition.model.StartSegmentDetectionRequest;
import com.amazonaws.services.rekognition.model.StartSegmentDetectionResult;
import com.amazonaws.services.rekognition.model.StartShotDetectionFilter;
import
 com.amazonaws.services.rekognition.model.StartTechnicalCueDetectionFilter;
import com.amazonaws.services.rekognition.model.TechnicalCueSegment;
import com.amazonaws.services.rekognition.model.AudioMetadata;
```

2. Agregue el siguiente código a la clase `VideoDetect`.

```
    //Copyright 2020 Amazon.com, Inc. or its affiliates. All Rights
 Reserved.
    //PDX-License-Identifier: MIT-0 (For details, see https://github.com/
awsdocs/amazon-rekognition-developer-guide/blob/master/LICENSE-SAMPLECODE.)


    private static void StartSegmentDetection(String bucket, String video)
 throws Exception{

        NotificationChannel channel= new NotificationChannel()
                .withSNSTopicArn(snsTopicArn)
                .withRoleArn(roleArn);

        float minTechnicalCueConfidence = 80F;
        float minShotConfidence = 80F;

        StartSegmentDetectionRequest req = new
 StartSegmentDetectionRequest()
                .withVideo(new Video()
                        .withS3Object(new S3Object()
                                .withBucket(bucket)
                                .withName(video)))
                .withSegmentTypes("TECHNICAL_CUE" , "SHOT")
                .withFilters(new StartSegmentDetectionFilters()
                        .withTechnicalCueFilter(new
 StartTechnicalCueDetectionFilter()

 .withMinSegmentConfidence(minTechnicalCueConfidence))
                        .withShotFilter(new StartShotDetectionFilter()

 .withMinSegmentConfidence(minShotConfidence)))
                .withJobTag("DetectingVideoSegments")
                .withNotificationChannel(channel);
```

```
        StartSegmentDetectionResult startLabelDetectionResult =
 rek.startSegmentDetection(req);
        startJobId=startLabelDetectionResult.getJobId();

    }

    private static void GetSegmentDetectionResults() throws Exception{

        int maxResults=10;
        String paginationToken=null;
        GetSegmentDetectionResult segmentDetectionResult=null;
        Boolean firstTime=true;


        do {
            if (segmentDetectionResult !=null){
                paginationToken = segmentDetectionResult.getNextToken();
            }

            GetSegmentDetectionRequest segmentDetectionRequest= new
 GetSegmentDetectionRequest()
                    .withJobId(startJobId)
                    .withMaxResults(maxResults)
                    .withNextToken(paginationToken);

            segmentDetectionResult =
 rek.getSegmentDetection(segmentDetectionRequest);

            if(firstTime) {
                System.out.println("\nStatus\n------");
                System.out.println(segmentDetectionResult.getJobStatus());
                System.out.println("\nRequested features
\n------------------");
                 for (SegmentTypeInfo requestedFeatures :
 segmentDetectionResult.getSelectedSegmentTypes()) {
                    System.out.println(requestedFeatures.getType());
                }
                 int count=1;
                 List<VideoMetadata> videoMetaDataList =
 segmentDetectionResult.getVideoMetadata();
                 System.out.println("\nVideo Streams\n-------------");
                 for (VideoMetadata videoMetaData: videoMetaDataList) {
                     System.out.println("Stream: " + count++);
```

```
                    System.out.println("\tFormat: " +
videoMetaData.getFormat());
                    System.out.println("\tCodec: " +
videoMetaData.getCodec());
                    System.out.println("\tDuration: " +
videoMetaData.getDurationMillis());
                    System.out.println("\tFrameRate: " +
videoMetaData.getFrameRate());
                }


                List<AudioMetadata> audioMetaDataList =
segmentDetectionResult.getAudioMetadata();
                System.out.println("\nAudio streams\n-------------");

                count=1;
                for (AudioMetadata audioMetaData: audioMetaDataList) {
                    System.out.println("Stream: " + count++);
                    System.out.println("\tSample Rate: " +
audioMetaData.getSampleRate());
                    System.out.println("\tCodec: " +
audioMetaData.getCodec());
                    System.out.println("\tDuration: " +
audioMetaData.getDurationMillis());
                    System.out.println("\tNumber of Channels: " +
audioMetaData.getNumberOfChannels());
                }
                System.out.println("\nSegments\n--------");

                firstTime=false;
            }


            //Show segment information

            List<SegmentDetection> detectedSegments=
segmentDetectionResult.getSegments();

            for (SegmentDetection detectedSegment: detectedSegments) {

               if
(detectedSegment.getType().contains(SegmentType.TECHNICAL_CUE.toString()))
{
                   System.out.println("Technical Cue");
```

```
                    TechnicalCueSegment
segmentCue=detectedSegment.getTechnicalCueSegment();
                    System.out.println("\tType: " + segmentCue.getType());
                    System.out.println("\tConfidence: " +
segmentCue.getConfidence().toString());
                }
               if
(detectedSegment.getType().contains(SegmentType.SHOT.toString())) {
                    System.out.println("Shot");
                    ShotSegment
segmentShot=detectedSegment.getShotSegment();
                    System.out.println("\tIndex " +
segmentShot.getIndex());
                    System.out.println("\tConfidence: " +
segmentShot.getConfidence().toString());
                }
                long seconds=detectedSegment.getDurationMillis();
                System.out.println("\tDuration : " + Long.toString(seconds)
+ " milliseconds");
                System.out.println("\tStart time code: " +
detectedSegment.getStartTimecodeSMPTE());
                System.out.println("\tEnd time code: " +
detectedSegment.getEndTimecodeSMPTE());
                System.out.println("\tDuration time code: " +
detectedSegment.getDurationSMPTE());
                System.out.println();

            }

       } while (segmentDetectionResult !=null &&
segmentDetectionResult.getNextToken() != null);

   }
```

3. En la función `main`, reemplace las líneas:

```
       StartLabelDetection(amzn-s3-demo-bucket, video);

       if (GetSQSMessageSuccess()==true)
        GetLabelDetectionResults();
```

por:

```
        StartSegmentDetection(amzn-s3-demo-bucket, video);

        if (GetSQSMessageSuccess()==true)
         GetSegmentDetectionResults();
```

Java V2

```
//snippet-start:[rekognition.java2.recognize_video_text.import]
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.S3Object;
import software.amazon.awssdk.services.rekognition.model.NotificationChannel;
import software.amazon.awssdk.services.rekognition.model.Video;
import
 software.amazon.awssdk.services.rekognition.model.StartTextDetectionRequest;
import
 software.amazon.awssdk.services.rekognition.model.StartTextDetectionResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import
 software.amazon.awssdk.services.rekognition.model.GetTextDetectionResponse;
import
 software.amazon.awssdk.services.rekognition.model.GetTextDetectionRequest;
import software.amazon.awssdk.services.rekognition.model.VideoMetadata;
import software.amazon.awssdk.services.rekognition.model.TextDetectionResult;
import java.util.List;
//snippet-end:[rekognition.java2.recognize_video_text.import]

/**
* Before running this Java V2 code example, set up your development environment,
 including your credentials.
*
* For more information, see the following documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
*/
public class DetectVideoSegments {

 private static String startJobId ="";
 public static void main(String[] args) {
```

```
    final String usage = "\n" +
        "Usage: " +
        "   <bucket> <video> <topicArn> <roleArn>\n\n" +
        "Where:\n" +
        "   bucket - The name of the bucket in which the video is located (for
example, (for example, amzn-s3-demo-bucket). \n\n"+
        "   video - The name of video (for example, people.mp4). \n\n" +
        "   topicArn - The ARN of the Amazon Simple Notification Service
(Amazon SNS) topic. \n\n" +
        "   roleArn - The ARN of the AWS Identity and Access Management (IAM)
role to use. \n\n" ;

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

    String bucket = args[0];
    String video = args[1];
    String topicArn = args[2];
    String roleArn = args[3];

    Region region = Region.US_WEST_2;
    RekognitionClient rekClient = RekognitionClient.builder()
        .region(region)
        .credentialsProvider(ProfileCredentialsProvider.create("profile-name"))
        .build();

    NotificationChannel channel = NotificationChannel.builder()
        .snsTopicArn(topicArn)
        .roleArn(roleArn)
        .build();

    startTextLabels(rekClient, channel, bucket, video);
    GetTextResults(rekClient);
    System.out.println("This example is done!");
    rekClient.close();
}

// snippet-start:[rekognition.java2.recognize_video_text.main]
public static void startTextLabels(RekognitionClient rekClient,
                                   NotificationChannel channel,
                                   String bucket,
```

```
                                String video) {
    try {
        S3Object s3Obj = S3Object.builder()
            .bucket(bucket)
            .name(video)
            .build();

        Video vidOb = Video.builder()
            .s3Object(s3Obj)
            .build();

        StartTextDetectionRequest labelDetectionRequest =
StartTextDetectionRequest.builder()
            .jobTag("DetectingLabels")
            .notificationChannel(channel)
            .video(vidOb)
            .build();

        StartTextDetectionResponse labelDetectionResponse =
rekClient.startTextDetection(labelDetectionRequest);
        startJobId = labelDetectionResponse.jobId();

    } catch (RekognitionException e) {
        System.out.println(e.getMessage());
        System.exit(1);
    }
}

public static void GetTextResults(RekognitionClient rekClient) {

    try {
        String paginationToken=null;
        GetTextDetectionResponse textDetectionResponse=null;
        boolean finished = false;
        String status;
        int yy=0 ;

        do{
            if (textDetectionResponse !=null)
                paginationToken = textDetectionResponse.nextToken();

            GetTextDetectionRequest recognitionRequest =
GetTextDetectionRequest.builder()
                .jobId(startJobId)
```

```
                .nextToken(paginationToken)
                .maxResults(10)
                .build();

            // Wait until the job succeeds.
            while (!finished) {
                textDetectionResponse =
rekClient.getTextDetection(recognitionRequest);
                status = textDetectionResponse.jobStatusAsString();

                if (status.compareTo("SUCCEEDED") == 0)
                    finished = true;
                else {
                    System.out.println(yy + " status is: " + status);
                    Thread.sleep(1000);
                }
                yy++;
            }

            finished = false;

            // Proceed when the job is done - otherwise VideoMetadata is null.
            VideoMetadata videoMetaData=textDetectionResponse.videoMetadata();
            System.out.println("Format: " + videoMetaData.format());
            System.out.println("Codec: " + videoMetaData.codec());
            System.out.println("Duration: " + videoMetaData.durationMillis());
            System.out.println("FrameRate: " + videoMetaData.frameRate());
            System.out.println("Job");

            List<TextDetectionResult> labels=
textDetectionResponse.textDetections();
            for (TextDetectionResult detectedText: labels) {
                System.out.println("Confidence: " +
detectedText.textDetection().confidence().toString());
                System.out.println("Id : " +
detectedText.textDetection().id());
                System.out.println("Parent Id: " +
detectedText.textDetection().parentId());
                System.out.println("Type: " +
detectedText.textDetection().type());
                System.out.println("Text: " +
detectedText.textDetection().detectedText());
                System.out.println();
            }
```

```
        } while (textDetectionResponse !=null &&
 textDetectionResponse.nextToken() != null);

    } catch(RekognitionException | InterruptedException e) {
        System.out.println(e.getMessage());
        System.exit(1);
    }
 }
 // snippet-end:[rekognition.java2.recognize_video_text.main]
}
```

Python

1. Añada el código siguiente a la clase `VideoDetect` que ha creado en el paso 1.

```
# Copyright 2020 Amazon.com, Inc. or its affiliates. All Rights Reserved.
# PDX-License-Identifier: MIT-0 (For details, see https://github.com/
awsdocs/amazon-rekognition-developer-guide/blob/master/LICENSE-SAMPLECODE.)

    def StartSegmentDetection(self):

        min_Technical_Cue_Confidence = 80.0
        min_Shot_Confidence = 80.0
        max_pixel_threshold = 0.1
        min_coverage_percentage = 60

        response = self.rek.start_segment_detection(
            Video={"S3Object": {"Bucket": self.bucket, "Name": self.video}},
            NotificationChannel={
                "RoleArn": self.roleArn,
                "SNSTopicArn": self.snsTopicArn,
            },
            SegmentTypes=["TECHNICAL_CUE", "SHOT"],
            Filters={
                "TechnicalCueFilter": {
                    "BlackFrame": {
                        "MaxPixelThreshold": max_pixel_threshold,
                        "MinCoveragePercentage": min_coverage_percentage,
                    },
                    "MinSegmentConfidence": min_Technical_Cue_Confidence,
                },
                "ShotFilter": {"MinSegmentConfidence": min_Shot_Confidence},
```

```
            }
        )

        self.startJobId = response["JobId"]
        print(f"Start Job Id: {self.startJobId}")

    def GetSegmentDetectionResults(self):
        maxResults = 10
        paginationToken = ""
        finished = False
        firstTime = True

        while finished == False:
            response = self.rek.get_segment_detection(
                JobId=self.startJobId, MaxResults=maxResults,
NextToken=paginationToken
            )

            if firstTime == True:
                print(f"Status\n------\n{response['JobStatus']}")
                print("\nRequested Types\n---------------")
                for selectedSegmentType in response['SelectedSegmentTypes']:
                    print(f"\tType: {selectedSegmentType['Type']}")
                    print(f"\t\tModel Version:
{selectedSegmentType['ModelVersion']}")

                print()
                print("\nAudio metadata\n--------------")
                for audioMetadata in response['AudioMetadata']:
                    print(f"\tCodec: {audioMetadata['Codec']}")
                    print(f"\tDuration: {audioMetadata['DurationMillis']}")
                    print(f"\tNumber of Channels:
{audioMetadata['NumberOfChannels']}")
                    print(f"\tSample rate: {audioMetadata['SampleRate']}")
                print()
                print("\nVideo metadata\n--------------")
                for videoMetadata in response["VideoMetadata"]:
                    print(f"\tCodec: {videoMetadata['Codec']}")
                    print(f"\tColor Range: {videoMetadata['ColorRange']}")
                    print(f"\tDuration: {videoMetadata['DurationMillis']}")
                    print(f"\tFormat: {videoMetadata['Format']}")
                    print(f"\tFrame rate: {videoMetadata['FrameRate']}")
                    print("\nSegments\n--------")
```

```
                firstTime = False

            for segment in response['Segments']:

                if segment["Type"] == "TECHNICAL_CUE":
                    print("Technical Cue")
                    print(f"\tConfidence: {segment['TechnicalCueSegment']
['Confidence']}")
                    print(f"\tType: {segment['TechnicalCueSegment']
['Type']}")

                if segment["Type"] == "SHOT":
                    print("Shot")
                    print(f"\tConfidence: {segment['ShotSegment']
['Confidence']}")
                    print(f"\tIndex: " + str(segment["ShotSegment"]
["Index"]))

                print(f"\tDuration (milliseconds):
 {segment['DurationMillis']}")
                print(f"\tStart Timestamp (milliseconds):
 {segment['StartTimestampMillis']}")
                print(f"\tEnd Timestamp (milliseconds):
 {segment['EndTimestampMillis']}")

                print(f"\tStart timecode: {segment['StartTimecodeSMPTE']}")
                print(f"\tEnd timecode: {segment['EndTimecodeSMPTE']}")
                print(f"\tDuration timecode: {segment['DurationSMPTE']}")

                print(f"\tStart frame number {segment['StartFrameNumber']}")
                print(f"\tEnd frame number: {segment['EndFrameNumber']}")
                print(f"\tDuration frames: {segment['DurationFrames']}")

                print()

            if "NextToken" in response:
                paginationToken = response["NextToken"]
            else:
                finished = True
```

2. En la función `main`, reemplace las líneas:

```
    analyzer.StartLabelDetection()
    if analyzer.GetSQSMessageSuccess()==True:
```

```
        analyzer.GetLabelDetectionResults()
```

por:

```
    analyzer.StartSegmentDetection()
    if analyzer.GetSQSMessageSuccess()==True:
        analyzer.GetSegmentDetectionResults()
```

> **Note**
>
> Si ya ha ejecutado un ejemplo de vídeo distinto de Análisis de un vídeo almacenado en un bucket de Amazon S3 con Java o Python (SDK), el código que se va a reemplazar podría ser diferente.

3. Ejecute el código. Se muestra información sobre los segmentos detectados en el vídeo de entrada.

Detección de la pruebas de vida del rostro

Amazon Rekognition Face Liveness le ayuda a comprobar que un usuario que se está sometiendo a una verificación facial está físicamente presente frente a una cámara. Detecta los ataques simulados que se presentan ante una cámara o cuando se intenta esquivar una cámara. Los usuarios pueden comprobar la vitalidad de sus rostros haciéndose una breve selfie en vídeo en la que siguen una serie de instrucciones destinadas a comprobar su presencia.

La vitalidad del rostro se determina mediante un cálculo probabilístico y, a continuación, se obtiene una puntuación de confianza (entre 0 y 100) tras la comprobación. Cuanto más alta sea la puntuación, mayor será la confianza en que la persona que recibe el cheque está viva. Face Liveness también devuelve un fotograma, denominado imagen de referencia, que se puede utilizar para comparar y buscar rostros. Como ocurre con cualquier sistema basado en probabilidades, Face Liveness no puede garantizar resultados perfectos. Úselo junto con otros factores para tomar una decisión basada en el riesgo sobre la identidad personal de los usuarios.

Face Liveness utiliza varios componentes:

- AWS Amplify el SDK (React, Swift (iOS) y Android) con componentes FaceLivenessDetector
- AWS SDKs
- AWS Cloud APIs

Cuando configura su aplicación para que se integre con la característica Face Liveness, utilice las siguientes operaciones de API:

- CreateFaceLivenessSession- Inicia una sesión de Face Liveness, lo que permite utilizar el modelo de detección de Face Liveness en su aplicación. Devuelve un valor SessionId para la sesión creada. También te permite configurar tu ChallengePrefrence, para que puedas usar la FaceMovementChallenge opción.
- StartFaceLivenessSession- Llamado por AWS Amplify FaceLivenessDetector. Inicia una secuencia de eventos que contiene información sobre los eventos y atributos relevantes de la sesión actual.
- GetFaceLivenessSessionResults- Recupera los resultados de una sesión específica de Face Liveness, incluida la puntuación de confianza de Face Liveness, una imagen de referencia y las imágenes de auditoría.

Utilizará el SDK AWS Amplify para integrar la función Face Liveness con sus flujos de trabajo de verificación basada en rostros para aplicaciones web. Cuando los usuarios se incorporen a su aplicación o se autentiquen a través de ella, envíelos al flujo de trabajo Face Liveness Check en el Amplify SDK. El Amplify SDK gestiona la interfaz de usuario y los comentarios en tiempo real para los usuarios mientras capturan su selfie en vídeo.

Cuando FaceMovementAndLightChallenge el rostro del usuario se mueve hacia el óvalo que se muestra en su dispositivo, el SDK de Amplify muestra una secuencia de luces de colores en la pantalla. A continuación, transmite de forma segura el vídeo de la selfie a la nube APIs. Alternativamente, al usarlo FaceMovementChallenge, el rostro del usuario se mueve hacia el óvalo que se muestra en su dispositivo, pero no hay una secuencia de luces de colores. Si bien «FaceMovementAndLightChallenge» sigue siendo la mejor opción para maximizar la precisión, «FaceMovementChallenge» permite a los clientes dar prioridad a una velocidad más rápida en las comprobaciones de funcionamiento que a la precisión. Al seleccionar una de estas opciones, los clientes deberían tener en cuenta los requisitos de su caso de uso, incluidos los tipos de ataque esperados, las tasas de falsas aceptaciones y rechazos erróneos deseadas y comprobaciones adicionales, como la geolocalización (por ejemplo, basada en IP), los códigos de pase único (OTPs), etc. Los clientes deben tomar esta decisión después de probar el rendimiento de Liveness con varios umbrales de confianza en su contenido específico. Además, con ambos tipos de comprobación de actividad, los propietarios de las aplicaciones deberían implementar controles para proteger el dispositivo desde el que se envía la transmisión de vídeo. Una vez finalizado el análisis, recibirá lo siguiente en el backend:

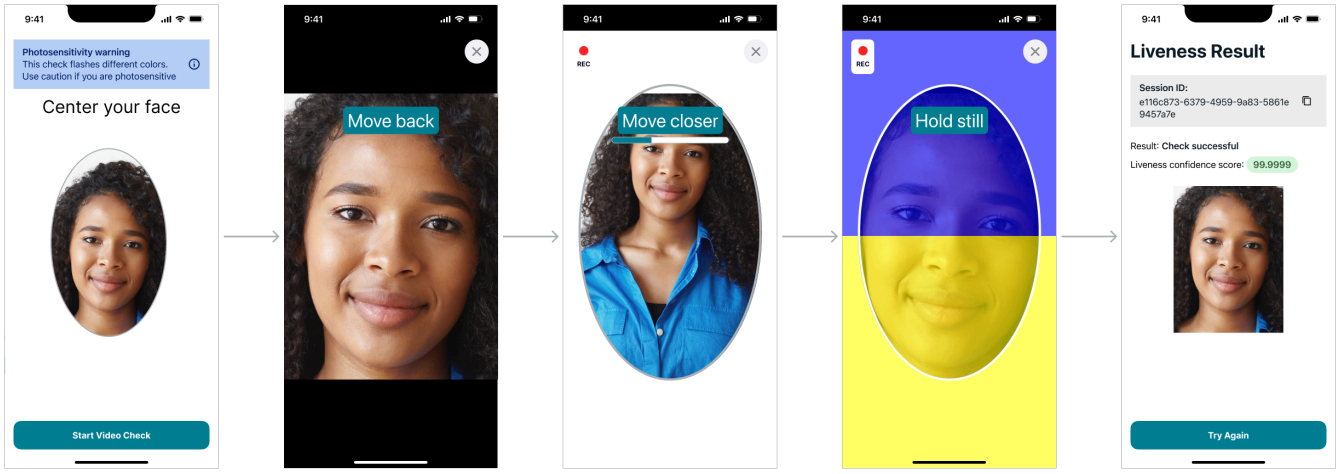
- Una puntuación de confianza en Face Liveness (entre 0 y 100)
- Una imagen de alta calidad llamada imagen de referencia que se puede utilizar para buscar rostros o buscar rostros
- Un conjunto de hasta cuatro imágenes, denominadas imágenes de auditoría, seleccionadas del vídeo de la selfie

Face Liveness se puede aprovechar para una variedad de casos de uso. Por ejemplo, Face Liveness se puede utilizar junto con la coincidencia de rostros (con CompareFacesy SearchFacesByImage) para verificar la identidad, estimar la edad en plataformas con restricciones de acceso basadas en la edad y detectar usuarios humanos reales y, al mismo tiempo, disuadir a los robots.

Puede obtener más información sobre los casos de uso a los que se destina el servicio, cómo utiliza este el machine learning (ML) y las consideraciones clave para el diseño y el uso responsables del servicio en la tarjeta de servicio de IA de Rekognition Face Liveness.

Puede establecer umbrales para Face Liveness y las puntuaciones de confianza en las coincidencias faciales. Los umbrales que elija deben reflejar su caso de uso. A continuación, envías una verificación de identidad approval/denial al usuario en función de si la puntuación está por encima o por debajo de los umbrales. Si se rechaza, pide al usuario que vuelva a intentarlo o le envía a otro método.

En el siguiente gráfico se muestra el flujo de usuarios, desde las instrucciones hasta la comprobación de la actividad y el resultado obtenido:



Requisitos de Face Liveness por parte del usuario

Amazon Rekognition Face Liveness requiere las siguientes especificaciones mínimas:

Dispositivos:

- El dispositivo debe tener una cámara frontal
- Frecuencia de actualización mínima de la pantalla del dispositivo: 60 Hz
- Tamaño mínimo de pantalla: 4 pulgadas
- El dispositivo no debe estar manipulado ni rooteado

Especificaciones de la cámara:

- Cámara a color: la cámara frontal debería poder grabar colores.
- Sin cámara virtual ni software de cámara.

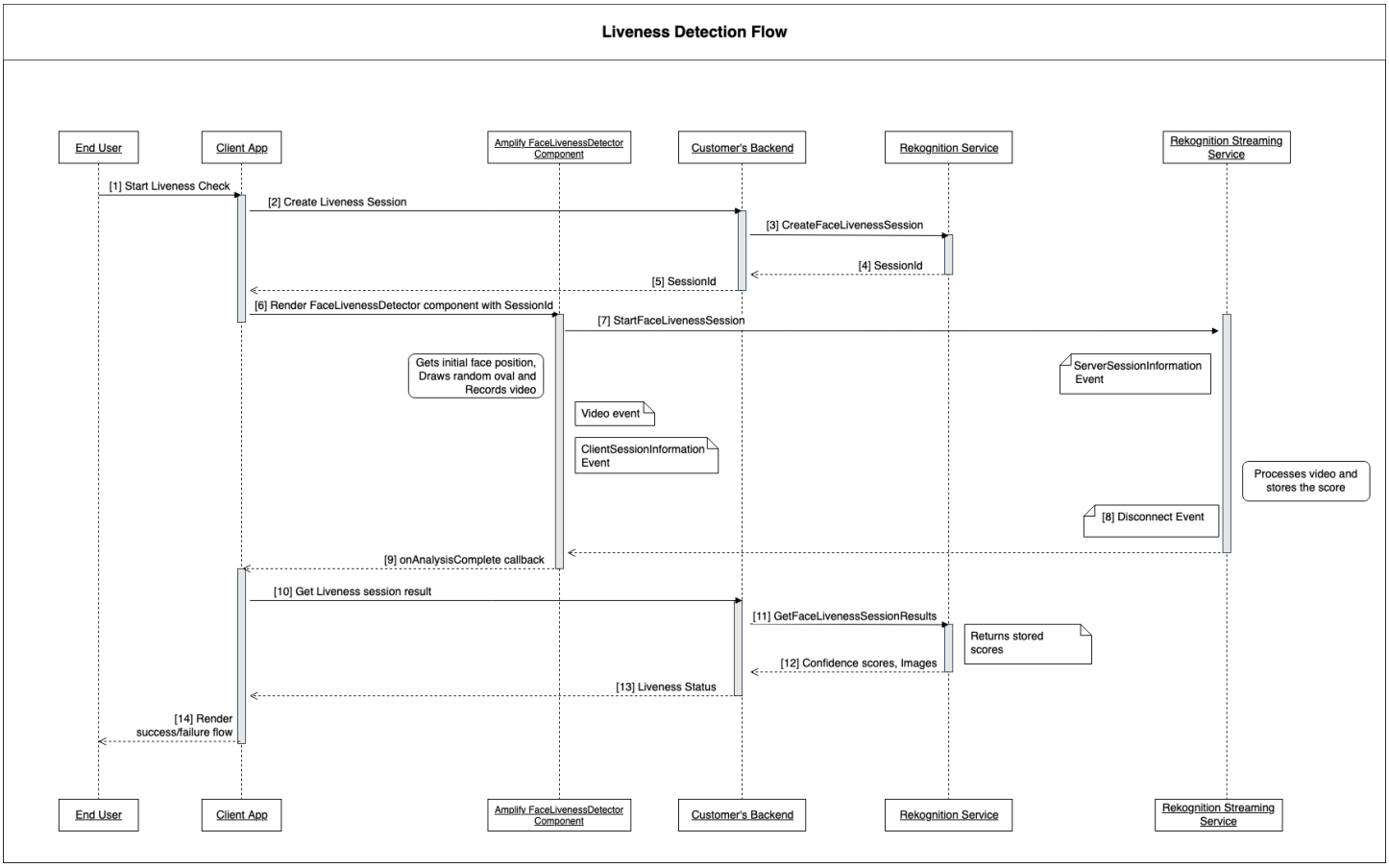
- Capacidad mínima de grabación: 15 fotogramas por segundo.
- Resolución mínima de grabación de vídeo: 480 x 640 px.
- Cuando los usuarios utilizan una cámara web con un ordenador de sobremesa para una comprobación de Face Liveness, es importante montar la cámara web en la parte superior de la misma pantalla en la que se inicia la comprobación de Face Liveness.

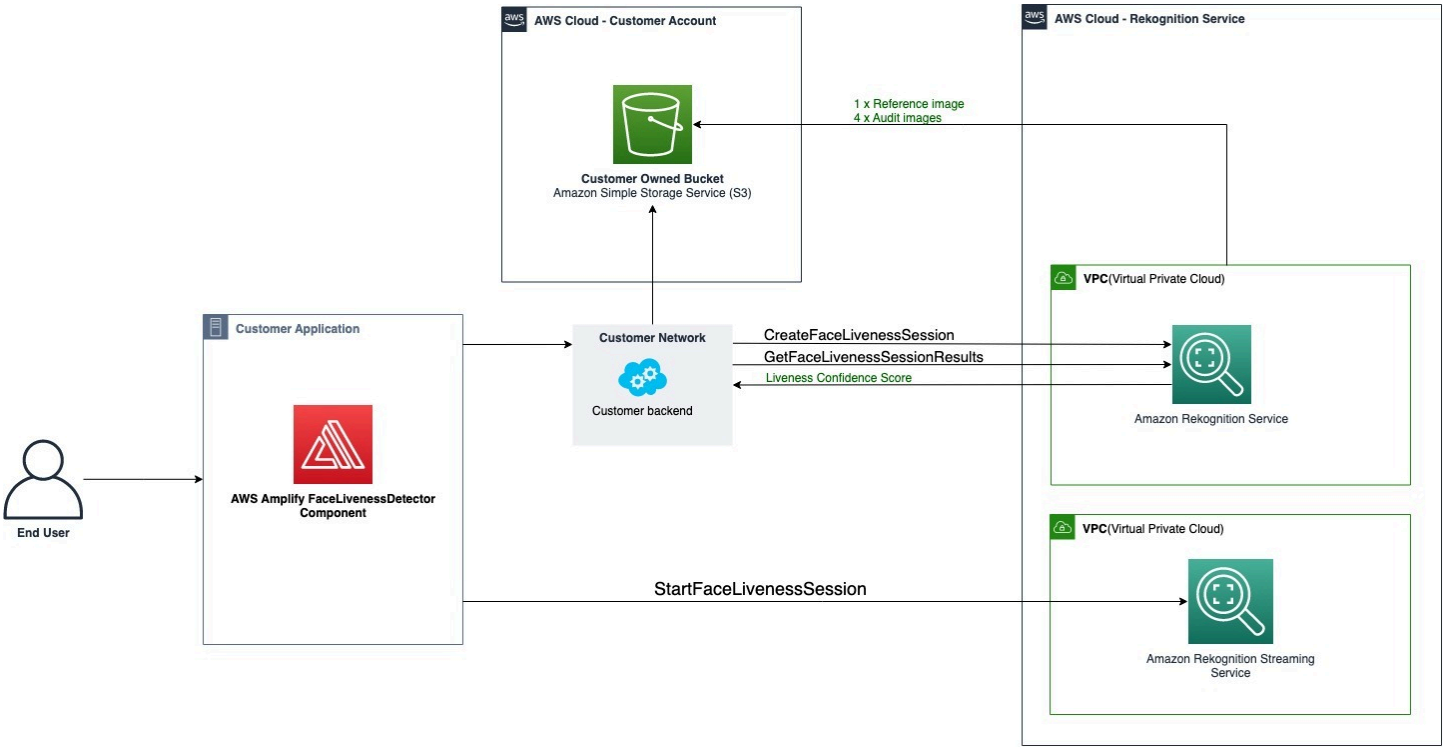
Ancho de banda mínimo requerido: 100 kbps

Navegadores compatibles: las tres últimas versiones de los principales navegadores, como Google Chrome, Mozilla Firefox, Apple Safari y Microsoft Edge. Para obtener más información acerca de los navegadores compatibles, consulte ¿Qué navegadores admite la Consola de administración de AWS?

Diagramas de arquitectura y secuencia

Los siguientes diagramas detallan el funcionamiento de Amazon Rekognition Face Liveness con respecto a la arquitectura y la secuencia de operaciones de la característica:





El proceso de verificación de Face Liveness consta de varios pasos, como se describe a continuación:

1. El usuario inicia una comprobación de Face Liveness en la aplicación cliente.
2. La aplicación cliente llama al backend del cliente, que a su vez llama al servicio Amazon Rekognition. El servicio crea una sesión de Face Liveness y devuelve una única. SessionId Nota: Una SessionId vez enviada, caduca en 3 minutos, por lo que solo hay un período de 3 minutos para completar los pasos 3 a 7 que se indican a continuación. Debe utilizarse un valor sessionID nuevo para cada comprobación de Face Liveness. Si se utiliza un valor sessionID dado para las siguientes comprobaciones de Face Liveness, estas generarán un error. Además, una SessionId caduca 3 minutos después de su envío, por lo que todos los datos de Liveness asociados a la sesión (p. ej., el ID de sesión, la imagen de referencia, las imágenes de auditoría, etc.) no estarán disponibles.
3. La aplicación cliente renderiza el componente FaceLivenessDetector Amplify utilizando las devoluciones de llamada SessionId obtenidas y apropiadas.
4. El FaceLivenessDetector componente establece una conexión con el servicio de streaming Amazon Rekognition, representa un óvalo en la pantalla del usuario y muestra una secuencia de luces de colores. FaceLivenessDetector graba y transmite vídeo en tiempo real al servicio de streaming Amazon Rekognition.

5. El servicio de streaming Amazon Rekognition procesa el vídeo en tiempo real, almacena los resultados y DisconnectEvent devuelve FaceLivenessDetector a al componente una vez finalizada la transmisión.
6. El FaceLivenessDetector componente `onAnalysisComplete` devuelve la llamada para indicar a la aplicación cliente que la transmisión ha finalizado y que las partituras están listas para su recuperación.
7. La aplicación cliente llama al servidor del cliente para obtener un indicador booleano que indica si el usuario estaba en directo o no. El backend del cliente realiza la solicitud al servicio de Amazon Rekognition para obtener la puntuación de confianza, la referencia y las imágenes de auditoría. El backend del cliente utiliza estos atributos para determinar si el usuario está activo y devuelve la respuesta adecuada a la aplicación del cliente.
8. Por último, la aplicación cliente pasa la respuesta al FaceLivenessDetector componente, que procesa el success/failure mensaje de forma adecuada para completar el flujo.

Requisitos previos

Los requisitos previos para usar Amazon Rekognition Face Liveness son los siguientes:

1. Configure una cuenta AWS
2. Configura Face Liveness AWS SDKs
3. Configura los recursos de AWS Amplify

Paso 1: Configurar una cuenta AWS

Si aún no tiene una AWS cuenta, complete los pasos que se indican en Crea una AWS cuenta y un usuario para crear una.

Paso 2: Configura la viveza facial AWS SDKs

Si aún no lo ha hecho, instale y configure el AWS CLI y el AWS SDKs. Para obtener más información, consulte Paso 2: Configure y AWS CLI AWS SDKs.

Hay varias formas de autenticar las llamadas al AWS SDK. En los ejemplos de esta guía se supone que se utiliza un perfil de credenciales predeterminado para llamar a los comandos de AWS CLI y a las operaciones de la API AWS del SDK.

Consulte la página Cómo conceder acceso mediante programación para obtener más información sobre cómo conceder a su cuenta de usuario el acceso al AWS SDK que elija. En la página también se explica cómo usar un perfil en el equipo local y cómo ejecutar el código de ejemplo en los AWS entornos.

Asegúrese de que el usuario que realiza las operaciones de Face Liveness tiene los permisos correctos para llamar a las operaciones, como los permisos `AmazonRekognitionFullAccess` y `AmazonS3FullAccess`.

Paso 3: Configurar AWS Amplify Resources

Para integrar Amazon Rekognition Face Liveness en su aplicación, debe configurar el SDK de Amplify para usar AWS el componente Amplify. FaceLivenessDetector

Si aún no lo ha hecho, siga las instrucciones para configurar la Interfaz de la línea de comandos de AWS (AWS CLI) en Comenzar a utilizar la AWS CLI. Una vez instalada la CLI, complete los pasos de configuración de la autenticación que se muestran en el sitio de documentos de la interfaz de usuario de Amplify para configurar los recursos de AWS Amplify.

Prácticas recomendadas para detectar la vitalidad del rostro

Es conveniente que siga varias prácticas recomendadas si va a utilizar Amazon Rekognition Face Liveness. Las prácticas recomendadas de Face Liveness incluyen pautas sobre dónde se deben realizar las comprobaciones de la vitalidad facial, el uso de imágenes de auditoría y la elección de los umbrales de confianza.

Consulte Recomendaciones para el uso de Face Liveness para ver la lista completa de las prácticas recomendadas.

Programación de Amazon Rekognition Face Liveness APIs

Para usar la API de Amazon Rekognition Face Liveness, debe crear un backend que lleve a cabo los siguientes pasos:

1. Llame CreateFaceLivenessSessionpara iniciar una sesión de Face Liveness. Cuando se completa la operación `CreateFaceLivenessSession`, la interfaz de usuario solicita al usuario que envíe una selfie en vídeo. A continuación, el FaceLivenessDetector componente de AWS Amplify llama StartFaceLivenessSessionpara realizar la detección de Liveness.

2. Llame GetFaceLivenessSessionResultspara obtener los resultados de detección asociados a una sesión de Face Liveness.
3. Continúe configurando su aplicación React para usar el FaceLivenessDetector componente siguiendo los pasos de la guía Amplify Liveness.

Antes de usar Face Liveness, asegúrese de haber creado una cuenta de AWS, configurar AWS CLI y AWS SDKs, y configurar AWS Amplify. También debe asegurarse de que la política de IAM de su API de backend tenga permisos que cubran lo siguiente: `GetFaceLivenessSessionResults` y `CreateFaceLivenessSession`. Consulte la sección Requisitos previos para obtener más información.

Paso 1: CreateFaceLivenessSession

CreateFaceLivenessSession La operación de la API crea una sesión de Face Liveness y devuelve una única`SessionId`.

Como parte de la entrada para esta operación, también es posible especificar la ubicación de un bucket de Amazon S3. Esto permite almacenar una imagen de referencia y las imágenes de auditoría generadas durante la sesión de Face Liveness. El bucket de Amazon S3 debe estar ubicado en la cuenta de AWS de la persona que llama y en la misma región que el punto de conexión de Face Liveness. Además, las claves de objeto de S3 las genera el sistema Face Liveness.

También es posible proporcionar un `AuditImagesLimit`, que es un número entre 0 y 4. De forma predeterminada, está establecido en 0. El número de imágenes devueltas es el máximo posible y se basa en la duración del vídeo de autorretrato.

Por último, puede proporcionar un desafío ChallengePreference, incluidos el tipo y la versión del desafío que desea que su sesión utilice de forma predeterminada. De forma predeterminada, se usará, FaceLivenessMovementAndLightChallenge pero se puede configurar en FaceMovementChallenge.

Ejemplo de solicitud

```
{
   "ClientRequestToken": "string",
   "KmsKeyId": "string",
   "Settings": {
      "AuditImagesLimit": number,
```

```
      "ChallengePreferences": [
         {
            "Type": "string",
            "Versions": {
               "Maximum": "string",
               "Minimum": "string"
            }
         }
      ],
      "OutputConfig": {
         "S3Bucket": "string",
         "S3KeyPrefix": "string"
      }
   }
}
```

Ejemplo de respuesta

```
{
    {"SessionId": "0f959dbb-37cc-45d8-a08d-dc42cce85fa8"}
}
```

Paso 2: StartFaceLivenessSession

Cuando finaliza la operación de CreateFaceLivenessSession API, el componente AWS Amplify realiza la operación de StartFaceLivenessSession API. Se le pide al usuario que capture un autorretrato en vídeo. Para que la comprobación se realice correctamente, el usuario debe colocar su cara dentro del óvalo que aparece en la pantalla y, al mismo tiempo, mantener una buena iluminación. Para obtener más información, consulte Recomendaciones para el uso de Face Liveness.

Esta operación de API requiere el vídeo capturado durante la sesión de Face Liveness, el sessionID obtenido de la operación de API y una devolución CreateFaceLivenessSession de llamada. `onAnalysisComplete` La devolución de llamada se puede utilizar para indicar al backend que llame a la operación de la GetFaceLivenessSessionResults API, lo que devuelve una puntuación de confianza, una referencia e imágenes de auditoría.

Tenga en cuenta que este paso lo realiza el FaceLivenessDetector componente AWS Amplify de la aplicación cliente. No necesita realizar una configuración adicional para llamar a `StartFaceLivenessSession`.

Paso 3: GetFaceLivenessSessionResults

La operación GetFaceLivenessSessionResults de la API recupera los resultados de una sesión específica de Face Liveness. Requiere el SessionId como entrada y devuelve la puntuación de confianza de Face Liveness correspondiente. También proporciona una imagen de referencia que incluye un recuadro delimitador de caras e imágenes de auditoría que también contienen recuadros delimitadores de caras. La puntuación de confianza de Face Liveness oscila entre 0 y 100.

Ejemplo de solicitud

```
{"SessionId": "0f959dbb-37cc-45d8-a08d-dc42cce85fa8"}
```

Ejemplo de respuesta

```
{
    "SessionId": "0f959dbb-37cc-45d8-a08d-dc42cce85fa8",
    "Confidence": 98.9735,
    "ReferenceImage": {
        "S3Object": {
            "Bucket": "s3-bucket-name",
            "Name": "file-name",
        },
        "BoundingBox": {
           "Height": 0.4943420886993408,
            "Left": 0.8435328006744385,
            "Top": 0.8435328006744385,
            "Width": 0.9521094560623169}
    },
    "AuditImages": [{
        "S3Object": {
            "Bucket": "s3-bucket-name",
            "Name": "audit-image-name",
        },
        "BoundingBox": {
           "Width": 0.6399999856948853,
           "Height": 0.47999998927116394,
           "Left": 0.1644444465637207,
           "Top": 0.17666666209697723}
    }],
```

```
    "Status": "SUCCEEDED"
}
```

Paso 4: Responder a los resultados

Tras la sesión de Face Liveness, compare la puntuación de confianza de la prueba con el umbral especificado. Si la puntuación es superior al umbral, el usuario puede pasar a la siguiente pantalla o tarea. Si la comprobación no se realiza correctamente, se notificará al usuario y se le pedirá que vuelva a intentarlo.

Llamando a la cara vivacidad APIs

Puede probar Amazon Rekognition Face Liveness con cualquier SDK AWS compatible, como el AWS Python SDK Boto3 o el AWS SDK for Java. Puede llamar al y con el `CreateFaceLivenessSession` SDK que elija. `GetFaceLivenessSessionResults` APIs La siguiente sección muestra cómo llamarlos APIs con Python y Java SDKs.

Para llamar a Face Liveness APIs:

- Si aún no lo ha hecho, debe crear o actualizar un usuario con permisos de `AmazonRekognitionFullAccess`. Para obtener más información, consulte Step 1: Set up an AWS account and create a User.
- Si aún no lo ha hecho, instale y configure la AWS CLI y AWS SDKs. Para obtener más información, consulte el paso 2: configurar la AWS CLI y AWS SDKs.

Python

El siguiente fragmento muestra cómo puede llamarlos APIs en sus aplicaciones de Python. Tenga en cuenta que para ejecutar este ejemplo necesitará usar al menos la versión 1.26.110 del SDK de Boto3, aunque se recomienda utilizar la versión más reciente del SDK.

```
import boto3

session = boto3.Session(profile_name='default')
client = session.client('rekognition')

def create_session():
```

```
    response = client.create_face_liveness_session()

    session_id = response.get("SessionId")
    print('SessionId: ' + session_id)

    return session_id


def get_session_results(session_id):

    response = client.get_face_liveness_session_results(SessionId=session_id)

    confidence = response.get("Confidence")
    status = response.get("Status")

    print('Confidence: ' + "{:.2f}".format(confidence) + "%")
    print('Status: ' + status)

    return status


def main():
    session_id = create_session()
    print('Created a Face Liveness Session with ID: ' + session_id)

    status = get_session_results(session_id)
    print('Status of Face Liveness Session: ' + status)


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

Java

El siguiente fragmento muestra cómo puede llamarlos APIs en sus aplicaciones Java:

```
package aws.example.rekognition.liveness;

import com.amazonaws.services.rekognition.AmazonRekognition;
import com.amazonaws.services.rekognition.AmazonRekognitionClientBuilder;
```

```
import com.amazonaws.services.rekognition.model.AmazonRekognitionException;
import com.amazonaws.services.rekognition.model.CreateFaceLivenessSessionRequest;
import com.amazonaws.services.rekognition.model.CreateFaceLivenessSessionResult;
import
 com.amazonaws.services.rekognition.model.GetFaceLivenessSessionResultsRequest;
import
 com.amazonaws.services.rekognition.model.GetFaceLivenessSessionResultsResult;

public class DemoLivenessApplication {

    static AmazonRekognition rekognitionClient;

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

        rekognitionClient = AmazonRekognitionClientBuilder.defaultClient();

        try {
            String sessionId = createSession();
            System.out.println("Created a Face Liveness Session with ID: " +
 sessionId);

            String status = getSessionResults(sessionId);
            System.out.println("Status of Face Liveness Session: " + status);

        } catch(AmazonRekognitionException e) {
           e.printStackTrace();
        }
    }

    private static String createSession() throws Exception {

        CreateFaceLivenessSessionRequest request = new
 CreateFaceLivenessSessionRequest();
        CreateFaceLivenessSessionResult result =
 rekognitionClient.createFaceLivenessSession(request);

        String sessionId = result.getSessionId();
        System.out.println("SessionId: " + sessionId);

        return sessionId;
    }

    private static String getSessionResults(String sessionId) throws Exception {
```

```
        GetFaceLivenessSessionResultsRequest request = new
 GetFaceLivenessSessionResultsRequest().withSessionId(sessionId);
        GetFaceLivenessSessionResultsResult result =
 rekognitionClient.getFaceLivenessSessionResults(request);

        Float confidence = result.getConfidence();
        String status = result.getStatus();

        System.out.println("Confidence: " + confidence);
        System.out.println("status: " + status);

        return status;
    }
}
```

Java V2

En el siguiente fragmento se muestra cómo llamar a Face Liveness APIs con el SDK de Java V2: AWS

```
package aws.example.rekognition.liveness;

import com.amazonaws.services.rekognition.AmazonRekognition;
import com.amazonaws.services.rekognition.AmazonRekognitionClientBuilder;
import com.amazonaws.services.rekognition.model.AmazonRekognitionException;
import com.amazonaws.services.rekognition.model.CreateFaceLivenessSessionRequest;
import com.amazonaws.services.rekognition.model.CreateFaceLivenessSessionResult;
import
 com.amazonaws.services.rekognition.model.GetFaceLivenessSessionResultsRequest;
import
 com.amazonaws.services.rekognition.model.GetFaceLivenessSessionResultsResult;

public class DemoLivenessApplication {

    static AmazonRekognition rekognitionClient;

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

        rekognitionClient = AmazonRekognitionClientBuilder.defaultClient();
```

```
        try {
            String sessionId = createSession();
            System.out.println("Created a Face Liveness Session with ID: " +
 sessionId);

            String status = getSessionResults(sessionId);
            System.out.println("Status of Face Liveness Session: " + status);

        } catch(AmazonRekognitionException e) {
           e.printStackTrace();
        }
    }

    private static String createSession() throws Exception {

        CreateFaceLivenessSessionRequest request = new
 CreateFaceLivenessSessionRequest();
        CreateFaceLivenessSessionResult result =
 rekognitionClient.createFaceLivenessSession(request);

        String sessionId = result.getSessionId();
        System.out.println("SessionId: " + sessionId);

        return sessionId;
    }

    private static String getSessionResults(String sessionId) throws Exception {

        GetFaceLivenessSessionResultsRequest request = new
 GetFaceLivenessSessionResultsRequest().withSessionId(sessionId);
        GetFaceLivenessSessionResultsResult result =
 rekognitionClient.getFaceLivenessSessionResults(request);

        Float confidence = result.getConfidence();
        String status = result.getStatus();

        System.out.println("Confidence: " + confidence);
        System.out.println("status: " + status);

        return status;
    }
}
```

Node.Js

En el siguiente fragmento se muestra cómo llamar a Face APIs Liveness con el SDK de Node.Js: AWS

```
const Rekognition = require("aws-sdk/clients/rekognition");

const rekognitionClient = new Rekognition({ region: "us-east-1" });

async function createSession() {
    const response = await rekognitionClient.createFaceLivenessSession().promise();

    const sessionId = response.SessionId;
    console.log("SessionId:", sessionId);

    return sessionId;
}

async function getSessionResults(sessionId) {
    const response = await rekognitionClient
        .getFaceLivenessSessionResults({
            SessionId: sessionId,
        })
        .promise();

    const confidence = response.Confidence;
    const status = response.Status;
    console.log("Confidence:", confidence);
    console.log("Status:", status);

    return status;
}

async function main() {
    const sessionId = await createSession();
    console.log("Created a Face Liveness Session with ID:", sessionId);

    const status = await getSessionResults(sessionId);
    console.log("Status of Face Liveness Session:", status);
}
```

```
main();
```

Node.Js (Javascript SDK v3)

En el siguiente fragmento se muestra cómo llamar a Face Liveness APIs con el SDK de Node.Js para Javascript v3: AWS

```
import { RekognitionClient, CreateFaceLivenessSessionCommand } from "@aws-sdk/
client-rekognition"; // ES Modules
import const { RekognitionClient, CreateFaceLivenessSessionCommand } =
 require("@aws-sdk/client-rekognition"); // CommonJS import
const client = new RekognitionClient(config);
const input = {
  KmsKeyId: "STRING_VALUE",
  Settings: {
    OutputConfig: { // LivenessOutputConfig
      S3Bucket: "STRING_VALUE", // required
      S3KeyPrefix: "STRING_VALUE",
    },
    AuditImagesLimit: Number("int"),
  },
  ClientRequestToken: "STRING_VALUE",
};
const command = new CreateFaceLivenessSessionCommand(input);
const response = await client.send(command);
// { // CreateFaceLivenessSessionResponse
//   SessionId: "STRING_VALUE", // required
// };
```

Configuración y personalización de la aplicación

Configuración de su aplicación

La aplicación Face Liveness puede funcionar en dispositivos móviles o navegadores web de escritorio. Querrá configurar los componentes de Face Liveness para que se integren con la solución que elija. También debe asegurarse de que su aplicación tenga permiso para usar la cámara de un dispositivo. La guía de Amplify Liveness proporciona instrucciones detalladas sobre cómo:

- Instalar y configurar AWS Amplify

- Importe y renderice el componente FaceLivenessDetector
- Escuchar las devoluciones de llamadas
- Ejemplo de mensaje de error de Render Amplify

Personalizar su aplicación

Puede personalizar determinados componentes de su aplicación de animación con AWS Amplify.

Para obtener información sobre la traducción, consulte la documentación de Amplify Authenticator.

Para obtener información sobre la personalización de los componentes y temas de Amplify, consulte la documentación de Amplify relativa a la creación de temas.

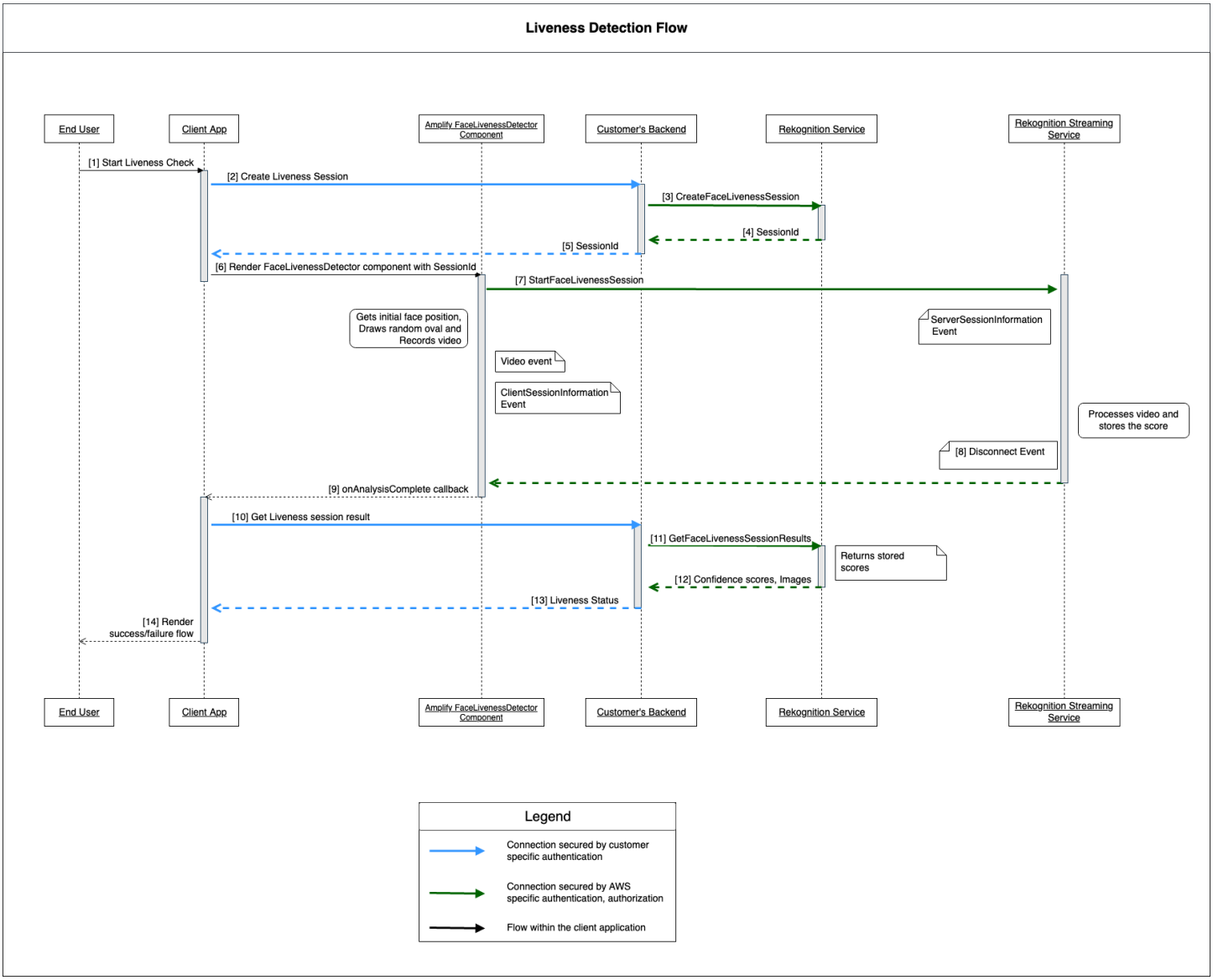
Modelo de responsabilidad compartida de Face Liveness

Los asuntos relacionados con la seguridad y la conformidad son una responsabilidad compartida entre AWS y el cliente. Obtenga más información sobre el modelo de responsabilidad compartida de AWS aquí.

1. Todas las llamadas al servicio de AWS (a través de la aplicación del cliente o del servidor del cliente) se autentican y autorizan con AWS Auth (AWS Authentication). Es responsabilidad de los propietarios del servicio Face Liveness garantizar que esto suceda.
2. Todas las llamadas al backend del cliente (desde la aplicación del cliente) se autentican y autorizan a través del cliente. Esta responsabilidad recae en el cliente. El cliente debe asegurarse de que las llamadas desde la aplicación cliente estén autenticadas y no hayan sido manipuladas de ninguna manera.
3. El backend del cliente debe identificar al usuario final que realiza el desafío Face Liveness. Es responsabilidad del cliente vincular a un usuario final a una sesión de Face Liveness. El servicio Face Liveness no distingue entre los usuarios finales. Solo puede identificar la identidad de AWS que llama (que gestiona el cliente).
4. AWS recomienda a los clientes aplicar comprobaciones de validación adicionales, como la geolocalización de la ubicación (por ejemplo, basada en IP), códigos One Time Pass (OTP), etc., además de Face Liveness, que se ajuste a los requisitos de sus casos de uso y a su posición de seguridad.

La configuración “FaceMovementAndLightChallenge” ofrece la máxima precisión para Rekognition Liveness, ya que requiere que los usuarios muevan la cara hacia la pantalla y se queden quietos durante una serie de luces parpadeantes. Recomendamos a los clientes que utilicen esta configuración predeterminada. Como alternativa, los clientes pueden activar la configuración “FaceMovementChallenge”, que reduce el tiempo de comprobación en varios segundos al eliminar las luces parpadeantes. Aunque “FaceMovementAndLightChallenge” sigue siendo la mejor opción para maximizar la precisión, “FaceMovementChallenge” permite a los clientes priorizar los controles de actividad más rápidos. Al seleccionar entre estas configuraciones, los clientes deberían tener en cuenta los requisitos de cada caso de uso, incluidos los tipos de ataque esperados y las tasas deseadas de falsas aceptaciones y falsos rechazos, y también deberían implementar comprobaciones adicionales, como la geolocalización (por ejemplo, basada en IP), los códigos One Time Pass (OTP), etc. Los clientes deben tomar esta decisión después de probar el rendimiento de Liveness con varios umbrales de confianza en función de su caso de uso. Los clientes son responsables de implementar controles para proteger el dispositivo desde el que se envía el vídeo.

El siguiente diagrama de flujo muestra qué llamadas autentican el servicio de AWS o el cliente:



Todas las llamadas al servicio Amazon Rekognition Face Liveness están protegidas por AWS Auth (mediante un mecanismo de firma de AWS). Estas incluyen las siguientes llamadas:

- [3] Llamada a la API CreateFaceLivenessSession (desde el backend del cliente)
- [7] Llamada a la API StartFaceLivenessSession (desde la aplicación del cliente)
- [11] Llamada a la API GetFaceLivenessSessionResults (desde el backend del cliente)

Todas las llamadas al backend del cliente deben tener un mecanismo de autenticación y autorización. Los clientes deben asegurarse de que el código/biblioteca/etc. de terceros utilizado se mantenga y desarrolle activamente. Los clientes también deben asegurarse de que el usuario final

correcto realiza las llamadas a la sesión correcta de Face Liveness. Los clientes deben autenticar y autorizar los siguientes flujos:

- [2] Cree una sesión de Face Liveness (desde la aplicación del cliente)
- [10] Obtenga el resultado de la sesión de Face Liveness (desde la aplicación del cliente)

Los clientes pueden seguir el modelo de seguridad STRIDE para asegurarse de que sus llamadas a la API estén protegidas.

Tipo	Descripción	Control de seguridad
Spoofing	Threat action aimed at accessing and use of another user's credentials, such as username and password.	Authentication
Tampering	Threat action intending to maliciously change or modify persistent data. Examples include records in a database, and the alteration of data in transit between two computers over an open network, such as the internet.	Integrity
Repudiation	Threat action aimed at performing prohibited operations in a system that lacks the ability to trace the operations.	Non-Repudiation
Information disclosure	Threat action intending to read a file that one was not granted access to, or to read data in transit.	Confidentiality
Denial of service	Threat action attempting to deny access to valid users,	Availability

	such as by making a web server temporarily unavailable or unusable.	
Elevation of privilege	Threat action intending to gain privileged access to resources in order to gain unauthorized access to information or to compromise a system.	Authorization

AWS protege sus conexiones de las siguientes maneras:

1. Calcule la firma de la solicitud y, a continuación, verifique la firma en el lado del servicio. Las solicitudes se autentican con esta firma.
2. Los clientes de AWS deben configurar los roles de IAM adecuados para autorizar determinadas acciones u operaciones. Estos roles de IAM son necesarios para realizar llamadas al servicio de AWS.
3. Solo se permiten las solicitudes HTTPS al servicio de AWS. Las solicitudes se cifran en la red abierta mediante TLS. Esto protege la confidencialidad de las solicitudes y mantiene su integridad.
4. El servicio de AWS registra datos suficientes para identificar las llamadas realizadas por los clientes. Esto evita los ataques de rechazo.
5. El servicio AWS es propietario y mantiene una disponibilidad suficiente

El cliente es responsable de proteger sus llamadas al servicio y a la API de las siguientes maneras:

1. El cliente debe asegurarse de seguir un mecanismo de autenticación adecuado. Existen varios mecanismos de autenticación que se pueden utilizar para autenticar una solicitud. Los clientes pueden explorar la autenticación basada en resúmenes, OAuth, OpenID Connect y otros mecanismos.
2. Los clientes deben asegurarse de que su servicio admite los canales de cifrado adecuados (como TLS/HTTPS) para realizar llamadas a la API del servicio.
3. Los clientes deben asegurarse de registrar los datos necesarios para identificar de forma exclusiva una llamada a la API y a la persona que llama. Deben poder identificar al cliente que llama a su API con parámetros definidos y la hora de las llamadas.

4. Los clientes deben asegurarse de que su sistema esté disponible y de que estén protegidos contra ataques DDoS. Estos son algunos ejemplos de técnicas de defensa contra los ataques DDoS.

Los clientes son responsables de mantener sus aplicaciones actualizadas. Para obtener más información, consulte Directrices de actualización de Face Liveness.

Directrices de actualización de Face Liveness

AWS actualiza periódicamente Face Liveness AWS SDKs (que se usa en el backend del cliente) y FaceLivenessDetector los componentes de AWS Amplify SDKs (que se usan en las aplicaciones de los clientes) para proporcionar nuevas funciones, seguridad mejorada y actualizada APIs, correcciones de errores, mejoras de usabilidad y más. Te recomendamos que mantengas esta opción SDKs up-to-date para garantizar un funcionamiento óptimo de la función. Si sigues utilizando versiones anteriores de SDKs, es posible que se bloqueen las solicitudes por motivos de mantenimiento y seguridad.

Face Liveness requiere que utilices el FaceLivenessDetector componente, incluido en AWS SDKs Amplify (React, iOS, Android).

Control de versiones y plazos

Estamos realizando el control de versiones de los siguientes componentes clave de la característica Face Liveness. Seguimos un formato de control de versiones semántico. Por ejemplo, un formato de versión de X.Y.Z donde X representa la versión principal, Y representa la versión secundaria y Z representa la versión del parche.

- Los desafíos de los usuarios de Face Liveness (por ejemplo, el FaceMovementAndLightChallenge desafío) forman parte de la API StartFaceLivenessSession
- FaceLivenessDetector los componentes suministrados a través de AWS Amplify SDKs se utilizan en aplicaciones cliente

Versiones principales: reservamos las actualizaciones de las versiones principales para actualizaciones críticas de seguridad, de última generación de API y de usabilidad espectaculares. Las aplicaciones y el backend del cliente deben actualizarse lo antes posible para que pueda seguir utilizando las característica de Face Liveness. Una vez que publiquemos una nueva versión principal, admitiremos la versión principal anterior durante 120 días a partir del día de publicación de la nueva

versión. Es posible que bloqueemos las solicitudes procedentes de la versión principal anterior después de 120 días.

Versiones secundarias: reservamos las actualizaciones de las versiones secundarias para características y mejoras importantes de seguridad y usabilidad. Recomendamos encarecidamente aplicar estas actualizaciones. Si bien nos esforzamos por garantizar que las actualizaciones menores sean compatibles con versiones anteriores durante el mayor tiempo posible, es posible que end-of-support anunciemos una versión secundaria anterior 180 días después del lanzamiento de una nueva versión secundaria.

Versiones de parches: reservamos las actualizaciones de las versiones de parches para corregir errores y realizar mejoras opcionales. Si bien le recomendamos que conserve su versión up-to-date para disfrutar de la mejor seguridad y experiencia de usuario, nos esforzamos por garantizar que las actualizaciones de los parches sean totalmente compatibles con versiones anteriores hasta que publiquemos una nueva versión principal o secundaria.

El período de control de versiones (120 días para la versión principal y 180 días para la versión secundaria) se aplica a la actualización del SDK de la aplicación, a la carga de la aplicación a la tienda de aplicaciones o al sitio web y a la descarga de la última versión de la aplicación por parte de los usuarios.

Matriz de versiones, lanzamientos y compatibilidad

El lanzamiento de una versión principal por un FaceLivenessDetector componente o por un problema de usuario suele coincidir. Para ayudarle a realizar un seguimiento de las dependencias de las versiones, consulte los recursos enlazados en las siguientes tablas.

Versiones y registros de cambios del SDK:

FaceLivenessDetector para el SDK web		FaceLivenessDetector para iOS SDK	FaceLivenessDetector para el SDK de Android
Versión actual	registro de cambios	Versión actual/registro de cambios	Versión actual/registro de cambios

Desafíos de los usuarios:

Nombre del desafío	Versión	Fecha de lanzamiento	Fecha de jubilación
FaceMovementAndLightChallenge	v1.0.0	10/04/2023	N/A
FaceMovementChallenge	v1.0.0	30 de abril de 2025	N/A

Comunicación de las nuevas versiones

AWS comunica los nuevos lanzamientos a través de los siguientes canales:

- Se envían notificaciones por correo electrónico sobre las actualizaciones del estado del servicio al correo electrónico de la cuenta asociado al ID de cuenta de Face Liveness.
- Publicó las actualizaciones AWS SDKs y las notificaciones asociadas en los GitHub repositorios respectivos.
- Publicó actualizaciones para AWS Amplify SDKs y las notificaciones asociadas en los repositorios respectivos GitHub .

Te recomendamos que te suscribas a estos canales para quedarte. up-to-date

Preguntas frecuentes sobre Face Liveness

Utilice las siguientes preguntas frecuentes para encontrar respuestas a las preguntas más frecuentes sobre Rekognition Face Liveness.

- ¿Cuáles son los resultados de una prueba de vitalidad facial?

  Rekognition Face Liveness proporciona los siguientes resultados para cada comprobación de vitalidad:

  - Puntuación de confianza: se devuelve una puntuación numérica que va de 0 a 100. Esta puntuación indica la probabilidad de que el vídeo de autorretrato sea de una persona real y no de un actor malintencionado que utilice una suplantación de identidad.

  - Imagen de alta calidad: se extrae una única imagen de alta calidad del vídeo de autorretrato. Este fotograma se puede utilizar para diversos fines, como la comparación de rostros, la estimación de la edad o la búsqueda de rostros.
  - Imágenes de auditoría: se obtienen hasta cuatro imágenes del vídeo de autorretrato, que se pueden utilizar como registro de auditoría.
- ¿Rekognition Face Liveness cumple con las pruebas de detección de ataques de presentación (PAD) de iBeta?

  Las pruebas de detección de ataques (PAD) de presentación de iBeta Quality Assurance se llevan a cabo de acuerdo con la norma ISO/IEC 30107-3. iBETA está acreditada por el NIST/NVLAP para realizar pruebas y proporcionar resultados según esta norma PAD. Rekognition Face Liveness superó las pruebas de conformidad de detección de ataques de presentación (PAD) iBeta de nivel 1 y 2 con una puntuación PAD perfecta. El informe puede consultarse en la página web de iBeta aquí.
- ¿Cómo puedo obtener un marco de alta calidad y marcos adicionales?

  El marco de alta calidad y los marcos adicionales pueden devolverse como bytes sin procesar o cargarse en un bucket de Amazon S3 que especifique, en función de las configuraciones de su solicitud de CreateFaceLivenessSessionAPI.
- ¿Puedo cambiar la ubicación del óvalo y las luces de colores?

  No. La ubicación del óvalo y las luces de colores son características que permiten una mayor precisión y, por lo tanto, no se pueden personalizar.
- ¿Puedo personalizar la interfaz de usuario según nuestra aplicación?

  Sí, puede personalizar la mayoría de los componentes de la pantalla, como el tema, el color, el idioma, el contenido del texto y la fuente, para adaptarlos a su aplicación. Encontrará detalles sobre cómo personalizar estos componentes en la documentación de nuestros componentes de interfaz de usuario de React, Swift y Android.
- ¿Puedo personalizar la hora y el tiempo de la cuenta atrás para poner una cara en la forma ovalada?

No, el tiempo de cuenta atrás y el tiempo de ajuste facial se han predeterminado a partir de estudios internos a gran escala realizados con miles de usuarios, con el objetivo de proporcionar un equilibrio óptimo entre mayor precisión y latencia. Por este motivo, estos ajustes de tiempo no se pueden personalizar.

- ¿Las luces de diferentes colores cumplen con las normas de accesibilidad?

  Sí, las luces de diferentes colores de nuestro producto cumplen con las pautas de accesibilidad descritas en las WCAG 2.1. Como se ha comprobado con más de 1000 controles de usuario, la experiencia de usuario muestra aproximadamente dos colores por segundo, lo que cumple con la recomendación de limitar los colores a tres por segundo. Esto reduce la probabilidad de desencadenar ataques epilépticos en la mayoría de la población.

- ¿El SDK ajusta el brillo de la pantalla para obtener resultados óptimos?

  El móvil Face Liveness SDKs (para Android e iOS) ajusta automáticamente el brillo cuando se inicia la comprobación. Sin embargo, en el caso del SDK web, las páginas web tienen limitaciones que impiden el ajuste automático del brillo. En estos casos, esperamos que la aplicación web indique a los usuarios finales que aumenten manualmente el brillo de la pantalla para obtener resultados óptimos.

- ¿Tiene que ser un óvalo? ¿Podríamos usar otras formas similares?

  No, el tamaño, la forma y la ubicación del óvalo no se pueden personalizar. El diseño ovalado específico se ha elegido cuidadosamente por su eficacia a la hora de capturar y analizar con precisión los movimientos faciales. Por lo tanto, la forma ovalada no se puede modificar.

- ¿Qué es la end-to-end latencia?

  Medimos la end-to-end latencia desde el momento en que el usuario inicia la acción necesaria para completar la comprobación de actividad hasta el momento en que el usuario obtiene el resultado (aprobado o rechazado). En el mejor de los casos, la latencia es de 5 segundos. En promedio, esperamos que sea de unos 7 segundos. En el peor de los casos, la latencia es de 11 segundos. Observamos que la end-to-end latencia varía en función de: el tiempo que tarda el usuario en completar la acción requerida (es decir, mover la cara hacia el óvalo), la conectividad de la red, la latencia de la aplicación, etc.

- ¿Puedo usar la característica Face Liveness sin Amplify SDK?

No, se requiere Amplify SKD para usar la característica Rekognition Face Liveness.

- ¿Dónde puedo encontrar los estados de error asociados a Face Liveness?

  Puede ver los diferentes estados de error de Face Liveness aquí.

- Face Liveness no está disponible en mi región. ¿Cómo puedo usar la característica?

  Puede elegir llamar a Face Liveness en cualquiera de las regiones en las que esté disponible, en función del tráfico y de la proximidad. Actualmente, Face Liveness está disponible en las siguientes AWS regiones:

  - Este de EE. UU. (Norte de Virginia)
  - Oeste de EE. UU. (Oregón)
  - Europa (Irlanda)
  - Asia-Pacífico (Tokio, Bombay)

  Aunque tu AWS cuenta esté ubicada en una región diferente, no se espera que la diferencia de latencia sea significativa. Puede obtener marcos para selfies e imágenes de auditoría de alta calidad a través de la ubicación de Amazon S3 o como bytes sin procesar, pero su bucket de Amazon S3 debe coincidir con la AWS región de Face Liveness. Si son diferentes, debe recibir las imágenes como bytes sin procesar.

- ¿Amazon Rekognition Liveness Detection utiliza el contenido de los clientes para mejorar el servicio?

  Puede optar por que sus imágenes y vídeos no se utilicen para mejorar o desarrollar la calidad de Rekognition y otras tecnologías de machine learning inteligencia artificial de Amazon mediante la política de cancelación de AWS Organizations. Para obtener información sobre cómo cancelar, consulte Administración de la política de cancelación de servicios de IA.

Análisis masivo

Amazon Rekognition Bulk Analysis le permite procesar una gran colección de imágenes de forma asíncrona mediante un archivo de manifiesto con la operación. StartMediaAnalysisJob El resultado de cada imagen individual coincide con el resultado devuelto por la operación que utilice para el análisis.

Actualmente, Rekognition admite el análisis con la operación. DetectModerationLabels

Se le cobrará por la cantidad de imágenes que el trabajo haya procesado correctamente. Los resultados de un trabajo finalizado se envían a un bucket de Amazon S3 especificado.

Tenga en cuenta que Análisis masivo no admite la integración de Amazon A2I.

La API puede detectar tipos de contenido animado o ilustrado y la información sobre el tipo de contenido detectado se devuelve como parte de la respuesta.

Temas

- Procesamiento de imágenes de forma masiva
- Manifiestos de salida de Análisis masivo
- Tipo de contenido
- Verificación de predicciones y entrenamiento de adaptadores

Procesamiento de imágenes de forma masiva

Puede iniciar un nuevo trabajo de análisis masivo enviando un archivo de manifiesto y llamando a la operación. StartMediaAnalysisJob El archivo de manifiesto de entrada contiene referencias a imágenes de un bucket de Amazon S3 y tiene el siguiente formato:

```
{"source-ref": "s3://foo/bar/1.jpg"}
```

Para crear un trabajo de análisis masivo (CLI)

1. Si aún no lo ha hecho:

a. Cree o actualice un usuario con los permisos `AmazonRekognitionFullAccess` y `AmazonS3ReadOnlyAccess`. Para obtener más información, consulte Paso 1: Configurar una cuenta de AWS y crear un usuario.

b. Instale y configure el AWS CLI y el AWS SDKs. Para obtener más información, consulte Paso 2: Configure y AWS CLI AWS SDKs.

2. Suba una imagen en su bucket de S3.

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

3. Utilice los siguientes comandos para crear y recuperar trabajos de análisis masivos.

CLI

Utilice el siguiente comando para llamar a la StartMediaAnalysisJoboperación y analizarla junto con la DetectModerationLabels operación:

```
# Requests
# Starting DetectModerationLabels job with default settings
aws rekognition start-media-analysis-job \
--operations-config "DetectModerationLabels={MinConfidence='1'}" \
--input "S3Object={Bucket=amzn-s3-demo-source-bucket,Name=my-input.jsonl}" \
--output-config "S3Bucket=amzn-s3-demo-destination-bucket;,S3KeyPrefix=my-results"
```

Puede obtener información sobre un trabajo determinado, como la ruta de Amazon S3 del depósito donde se almacenan los resultados y los archivos de resumen, mediante la GetMediaAnalysisJoboperación. Le proporciona un identificador de trabajo devuelto por StartMediaAnalysisJob o ListMediaAnalysisJob. Los detalles de los trabajos individuales solo se retienen durante un año.

```
# Request
aws rekognition get-media-analysis-job \
--job-id customer-job-id
```

Puede enumerar todos sus análisis masivos mediante la operación de ListMediaAnalysisJobstrabajo, que devuelve páginas de trabajos. Con el argumento `max-results`, puede especificar el número máximo de trabajos que se van a devolver por página,

limitado al valor de `max-results`. Se devuelven un máximo de 100 resultados por página. Los detalles de los trabajos individuales solo se retienen durante un año.

```
# Request
# Specify number of jobs to return per page, limited to max-results.
aws rekognition list-media-analysis-jobs --max-results 1
```

Manifiestos de salida de Análisis masivo

El trabajo de análisis masivo genera un archivo de manifiesto de salida que contiene los resultados del trabajo, así como un resumen del manifiesto que contiene estadísticas y detalles sobre cualquier error al procesar las entradas del manifiesto de entrada.

Si se incluyeron entradas duplicadas en el manifiesto de entrada, el trabajo no intentará filtrar las entradas únicas, sino que procesará todas las entradas proporcionadas.

El archivo de manifiesto de salida tiene el siguiente formato:

```
// Output manifest for content moderation
{"source-ref":"s3://foo/bar/1.jpg", "detect-moderation-labels":
 {"ModerationLabels":[],"ModerationModelVersion":"7.0","ContentTypes":
[{"Confidence":72.7257,"Name":"Animated"}]}}
```

El resumen del manifiesto de salida tiene el siguiente formato:

```
{
    "version": "1.0",               # Schema version, 1.0 for GA.
    "statistics": {
        "total-json-lines": Number,  # Total number json lines (images) in the input
 manifest.
        "valid-json-lines": Number,  # Total number of JSON Lines (images) that contain
 references to valid images.
        "invalid-json-lines": Number # Total number of invalid JSON Lines. These lines
 were not handled.
    },
    "errors": [
        {
            "line-numer": Number,   # The number of the line in the manifest where the
 error occured.
            "source-ref": "String", # Optional. Name of the file if was parsed.
```

```
            "code": "String",       # Error code.
            "message": "String"     # Description of the error.
        }
    ]
}
```

Tipo de contenido

La operación devuelve la información sobre el tipo de contenido multimedia analizado por StartMediaAnalysisJob operación. GetMediaAnalysisJob ContentType puede ser una de dos categorías diferentes:

- Contenido animado, que incluye videojuegos y animaciones (por ejemplo, dibujos animados, cómics, manga o anime).
- Contenido ilustrado, que incluye dibujos, pinturas y bocetos.

Verificación de predicciones y entrenamiento de adaptadores

El análisis masivo también se puede aprovechar a través de la consola Rekognition para obtener predicciones para un lote de imágenes, verificar estas predicciones y, a continuación, crear un adaptador con las predicciones verificadas. Los adaptadores le permiten mejorar la precisión de cualquier operación de Rekognition compatible.

Actualmente, puede crear adaptadores para usarlos con la característica de moderación personalizada de Rekognition. Al crear un adaptador y proporcionárselo a la DetectModerationLabelsoperación, puede lograr una mayor precisión en las tareas de moderación de contenido relacionadas con su caso de uso específico.

Para obtener más información acerca de la Moderación personalizada, consulte Mejora de la precisión con la moderación personalizada. Consulte Análisis y verificación masivos para obtener una explicación sobre cómo verificar las predicciones realizadas con el análisis masivo. Para ver un tutorial sobre cómo usar la consola Rekognition para verificar las predicciones y crear un adaptador, consulte Tutorial sobre el adaptador de moderación personalizado.

Tutoriales

Estos tutoriales multiservicio muestran cómo utilizar las operaciones de la API de Rekognition junto con otros AWS servicios para crear aplicaciones de muestra y realizar diversas tareas. La mayoría de estos tutoriales utilizan Amazon S3 para almacenar imágenes o vídeos. Otros servicios de uso común incluyen AWS Lambda.

Temas

- Almacenamiento de datos de Amazon Rekognition con Amazon RDS y DynamoDB
- Uso de Amazon Rekognition y Lambda para etiquetar activos en un bucket de Amazon S3
- Creación de aplicaciones de análisis de vídeo de AWS
- Creación de una función de Lambda de Amazon Rekognition
- Uso de Amazon Rekognition para la verificación de identidad
- Uso de Amazon Rekognition (REK) para detectar etiquetas de aplicaciones de marketing
- Detección de etiquetas en una imagen mediante Lambda y Python

Almacenamiento de datos de Amazon Rekognition con Amazon RDS y DynamoDB

Al utilizar Amazon Rekognition APIs, es importante recordar que las operaciones de la API no guardan ninguna de las etiquetas generadas. Puede guardar estas etiquetas colocándolas en la base de datos, junto con los identificadores de las imágenes respectivas.

En este tutorial se muestra cómo detectar etiquetas y cómo guardarlas en una base de datos. La aplicación de ejemplo desarrollada en este tutorial leerá imágenes de un bucket de Amazon S3, realizará la DetectLabelsoperación en estas imágenes y almacenará las etiquetas resultantes en una base de datos. La aplicación almacenará los datos en una instancia de base de datos de Amazon RDS o en una base de datos de DynamoDB, según el tipo de base de datos que desee utilizar.

Usará el AWS SDK para Python en este tutorial. También puedes consultar el GitHub repositorio de ejemplos del SDK de AWS documentación para ver más tutoriales de Python.

Temas

- Requisitos previos
- Cómo obtener etiquetas para imágenes en bucket de Amazon S3

- Creación de una tabla de Amazon DynamoDB
- Carga de datos a DynamoDB
- Creación de una base de datos de MySQL en Amazon RDS
- Carga de datos a una tabla MySQL de Amazon RDS

Requisitos previos

Antes de comenzar este tutorial, necesitará instalar Python y completar los pasos necesarios para configurar el AWS SDK de Python. Además de esto, asegúrese de que:

Ha creado una cuenta de AWS y un rol de IAM

Ha instalado el SDK de Python (Boto3)

Has configurado correctamente tus credenciales de AWS acceso

Ha creado un bucket de Amazon S3 y lo ha llenado de imágenes

Ha creado una instancia de base de datos de RDS, si utiliza RDS para almacenar datos

Cómo obtener etiquetas para imágenes en bucket de Amazon S3

Comience por escribir una función que tome el nombre de una imagen de su bucket de Amazon S3 y recupere esa imagen. Esta imagen se mostrará para confirmar que se están transfiriendo las imágenes correctas a una llamada DetectLabelsque también está incluida en la función.

1. Busque el bucket de Amazon S3 que desee usar y anote su nombre. Realizará llamadas a este bucket de Amazon S3 y leerá las imágenes que contiene. Asegúrese de que su depósito contenga algunas imágenes para pasarlas a la DetectLabelsoperación.
2. Escriba el código para conectarse a su bucket de Amazon S3. Puede conectarse al recurso de Amazon S3 con Boto3 para recuperar una imagen de un bucket de Amazon S3. Una vez se haya conectado al recurso de Amazon S3, puede acceder a su bucket proporcionando el método del bucket con el nombre de su bucket de Amazon S3. Tras conectarse al bucket de Amazon S3, puede recuperar las imágenes del bucket mediante el método Object. Mediante Matplotlib, puede utilizar esta conexión para visualizar las imágenes a medida que se procesan. Boto3 también se utiliza para conectarse al cliente de Rekognition.

   En el siguiente código, introduzca su región en el parámetro region_name. Pasarás el nombre del bucket de Amazon S3 y el nombre de la imagen a DetectLabels, que devolverá las etiquetas

de la imagen correspondiente. Tras seleccionar solo las etiquetas de la respuesta, se devolverán tanto el nombre de la imagen como las etiquetas.

```
import boto3
from io import BytesIO
from matplotlib import pyplot as plt
from matplotlib import image as mp_img

boto3 = boto3.Session()

def read_image_from_s3(bucket_name, image_name):

    # Connect to the S3 resource with Boto 3
    # get bucket and find object matching image name
    s3 = boto3.resource('s3')
    bucket = s3.Bucket(name=bucket_name)
    Object = bucket.Object(image_name)

    # Downloading the image for display purposes, not necessary for detection of
 labels
    # You can comment this code out if you don't want to visualize the images
    file_name = Object.key
    file_stream = BytesIO()
    Object.download_fileobj(file_stream)
    img = mp_img.imread(file_stream, format="jpeg")
    plt.imshow(img)
    plt.show()

    # get the labels for the image by calling DetectLabels from Rekognition
    client = boto3.client('rekognition', region_name="region-name")
    response = client.detect_labels(Image={'S3Object': {'Bucket': bucket_name,
 'Name': image_name}},
                                    MaxLabels=10)

    print('Detected labels for ' + image_name)

    full_labels = response['Labels']

    return file_name, full_labels
```

3. Guarde este código en un archivo llamado get_images.py.

Creación de una tabla de Amazon DynamoDB

El código siguiente usa Boto3 para conectarse a DynamoDB y usa el método `CreateTable` de DynamoDB para crear una tabla denominada Images. La tabla tiene una clave primaria compuesta de una clave de partición denominada Image y de una clave de clasificación denominada Labels. La clave Image contiene el nombre de la imagen, mientras que la clave Labels almacena las etiquetas asignadas a esa imagen.

```
import boto3

def create_new_table(dynamodb=None):
    dynamodb = boto3.resource(
        'dynamodb',)
    # Table defination
    table = dynamodb.create_table(
        TableName='Images',
        KeySchema=[
            {
                'AttributeName': 'Image',
                'KeyType': 'HASH'  # Partition key
            },
            {
                'AttributeName': 'Labels',
                'KeyType': 'RANGE'  # Sort key
            }
        ],
        AttributeDefinitions=[
            {
                'AttributeName': 'Image',
                'AttributeType': 'S'
            },
            {
                'AttributeName': 'Labels',
                'AttributeType': 'S'
            },
        ],
        ProvisionedThroughput={
            'ReadCapacityUnits': 10,
            'WriteCapacityUnits': 10
        }
    )
    return table
```

```
if __name__ == '__main__':
    device_table = create_new_table()
    print("Status:", device_table.table_status)
```

Guarde este código en un editor y ejecútelo una vez para crear una tabla de DynamoDB.

Carga de datos a DynamoDB

Ahora que se ha creado la base de datos de DynamoDB y tiene una función para obtener etiquetas para las imágenes, puede almacenar las etiquetas en DynamoDB. El siguiente código recupera todas las imágenes de un bucket de S3, obtiene las etiquetas para ellas y almacena los datos en DynamoDB.

1. Deberá escribir el código para subir los datos en DynamoDB. Se utiliza una función llamada `get_image_names` para conectarse a su bucket de Amazon S3, que devuelve los nombres de todas las imágenes del bucket en forma de lista. Pasará esta lista a la función `read_image_from_S3`, que se importará del archivo `get_images.py` que creó.

   ```
   import boto3
   import json
   from get_images import read_image_from_s3

   boto3 = boto3.Session()

   def get_image_names(name_of_bucket):

       s3_resource = boto3.resource('s3')
       my_bucket = s3_resource.Bucket(name_of_bucket)
       file_list = []
       for file in my_bucket.objects.all():
           file_list.append(file.key)
       return file_list
   ```

2. La función `read_image_from_S3` que creamos anteriormente devolverá el nombre de la imagen que se está procesando y el diccionario de etiquetas asociadas a esa imagen. Se usa una función llamada `find_values` para obtener solo las etiquetas de la respuesta. A continuación, el nombre de la imagen y sus etiquetas estarán listos para subirse en la tabla de DynamoDB.

   ```
   def find_values(id, json_repr):
       results = []
   ```

```
    def _decode_dict(a_dict):
        try:
            results.append(a_dict[id])
        except KeyError:
            pass
        return a_dict

    json.loads(json_repr, object_hook=_decode_dict) # Return value ignored.
    return results
```

3. Utilizará una tercera función, llamada `load_data`, para subir las imágenes y etiquetas en la tabla de DynamoDB que ha creado.

```
def load_data(image_labels, dynamodb=None):

    if not dynamodb:
        dynamodb = boto3.resource('dynamodb')

    table = dynamodb.Table('Images')

    print("Adding image details:", image_labels)
    table.put_item(Item=image_labels)
    print("Success!!")
```

4. Aquí es donde se llaman las tres funciones que definimos anteriormente y se llevan a cabo las operaciones. Añada las tres funciones definidas anteriormente, junto con el código siguiente, a un archivo de Python. Ejecute el código.

```
bucket = "bucket_name"
file_list = get_image_names(bucket)

for file in file_list:
    file_name = file
    print("Getting labels for " + file_name)
    image_name, image_labels = read_image_from_s3(bucket, file_name)
    image_json_string = json.dumps(image_labels, indent=4)
    labels=set(find_values("Name", image_json_string))
    print("Labels found: " + str(labels))
    labels_dict = {}
    print("Saving label data to database")
    labels_dict["Image"] = str(image_name)
    labels_dict["Labels"] = str(labels)
```

```
    print(labels_dict)
    load_data(labels_dict)
    print("Success!")
```

Acabas de generar etiquetas para tus imágenes y las has almacenado en una instancia de DynamoDB. DetectLabels Asegúrese de eliminar todos los recursos que ha creado mientras sigue este tutorial. Eso evitará que le cobren por los recursos que no utilice.

Creación de una base de datos de MySQL en Amazon RDS

Antes de continuar, asegúrese de haber completado el procedimiento de configuración de Amazon RDS y de haber creado una instancia de base de datos de MySQL con Amazon RDS.

El siguiente código utiliza la biblioteca PyMySQL y su instancia de base de datos de Amazon RDS. Crea una tabla que contiene los nombres de las imágenes y las etiquetas asociadas a esas imágenes. Amazon RDS recibe comandos para crear tablas e insertar datos en ellas. Para usar Amazon RDS, debe conectarse al host de Amazon RDS con su nombre de host, nombre de usuario y contraseña. Para conectarse a Amazon RDS, debe proporcionar estos argumentos a la `connect` función de PyMy SQL y crear una instancia de un cursor.

1. En el siguiente código, sustituya el valor de host por su punto de conexión de host de Amazon RDS y sustituya el valor de user por el nombre de usuario maestro asociado a su instancia de Amazon RDS. También tendrá que reemplazar la contraseña por la contraseña maestra de su usuario principal.

   ```
   import pymysql

   host = "host-endpoint"
   user = "username"
   password = "master-password"
   ```

2. Cree una base de datos y una tabla para insertar sus datos de imagen y etiqueta. Para ello, ejecute y confirme una consulta de creación. El siguiente código crea una base de datos. Ejecute este código solo una vez.

   ```
   conn = pymysql.connect(host=host, user=user, passwd=password)
   print(conn)
   cursor = conn.cursor()
   print("Connection successful")
   ```

```
# run once
create_query = "create database rekogDB1"
print("Creation successful!")
cursor.execute(create_query)
cursor.connection.commit()
```

3. Una vez creada la base de datos, debe crear una tabla en la que insertar los nombres y etiquetas de las imágenes. Para crear una tabla, primero debe pasar el comando use de SQL, junto con el nombre de la base de datos, a la función `execute`. Una vez realizada la conexión, se ejecuta una consulta para crear una tabla. El siguiente código se conecta a la base de datos y, a continuación, crea una tabla con una clave principal, denominada `image_id`, y un atributo de texto que almacena las etiquetas. Utilice las importaciones y variables que definió anteriormente y ejecute este código para crear una tabla en la base de datos.

```
# connect to existing DB
cursor.execute("use rekogDB1")
cursor.execute("CREATE TABLE IF NOT EXISTS test_table(image_id VARCHAR (255)
 PRIMARY KEY, image_labels TEXT)")
conn.commit()
print("Table creation - Successful creation!")
```

Carga de datos a una tabla MySQL de Amazon RDS

Tras crear la base de datos de Amazon RDS y una tabla en la base de datos, puede obtener etiquetas para sus imágenes y almacenarlas en la base de datos de Amazon RDS.

1. Conéctese a su bucket de Amazon S3 y recupere los nombres de todas las imágenes del bucket. Estos nombres de imágenes se pasarán a la función `read_image_from_s3` que creó anteriormente para obtener las etiquetas de todas sus imágenes. El siguiente código se conecta a su bucket de Amazon S3 y devuelve una lista de todas las imágenes de su bucket.

```
import pymysql
from get_images import read_image_from_s3
import json
import boto3

host = "host-endpoint"
user = "username"
password = "master-password"
```

```
conn = pymysql.connect(host=host, user=user, passwd=password)
print(conn)
cursor = conn.cursor()
print("Connection successful")

def get_image_names(name_of_bucket):

    s3_resource = boto3.resource('s3')
    my_bucket = s3_resource.Bucket(name_of_bucket)
    file_list = []
    for file in my_bucket.objects.all():
        file_list.append(file.key)
    return file_list
```

2. La respuesta de la DetectLabelsAPI contiene algo más que las etiquetas, así que escriba una función para extraer únicamente los valores de las etiquetas. La siguiente función devuelve una lista que únicamente contiene las etiquetas.

```
def find_values(id, json_repr):
    results = []

    def _decode_dict(a_dict):
        try:
            results.append(a_dict[id])
        except KeyError:
            pass
        return a_dict

    json.loads(json_repr, object_hook=_decode_dict) # Return value ignored.
    return results
```

3. Necesitará una función para insertar los nombres y etiquetas de las imágenes en la tabla. La siguiente función ejecuta una consulta de inserción e inserta cualquier par de nombres y etiquetas de imagen determinados.

```
def upload_data(image_id, image_labels):

    # insert into db
    cursor.execute("use rekogDB1")
    query = "INSERT IGNORE INTO test_table(image_id, image_labels) VALUES (%s, %s)"
    values = (image_id, image_labels)
    cursor.execute(query, values)
```

```
    conn.commit()
    print("Insert successful!")
```

4. Por último, debe ejecutar las funciones que ha definido anteriormente. En el siguiente código, se recopilan los nombres de todas las imágenes del depósito y se proporcionan a la función que las invoca DetectLabels. Después, las etiquetas y el nombre de la imagen a la que se aplican se cargan en la base de datos de Amazon RDS. Copie las tres funciones definidas anteriormente, junto con el código siguiente, a un archivo de Python. Ejecute el archivo Python.

```
bucket = "bucket-name"
file_list = get_image_names(bucket)

for file in file_list:
    file_name = file
    print("Getting labels for " + file_name)
    image_name, image_labels = read_image_from_s3(bucket, file_name)
    image_json = json.dumps(image_labels, indent=4)
    labels=set(find_values("Name", image_json))
    print("Labels found: " + str(labels))
    unique_labels=set(find_values("Name", image_json))
    print(unique_labels)
    image_name_string = str(image_name)
    labels_string = str(unique_labels)
    upload_data(image_name_string, labels_string)
    print("Success!")
```

Ha utilizado DetectLabels correctamente la generación de etiquetas para sus imágenes y las ha almacenado en una base de datos MySQL mediante Amazon RDS. Asegúrese de eliminar todos los recursos que ha creado mientras sigue este tutorial. Esto evitará que le cobren por los recursos que no utilice.

Para ver más ejemplos de AWS varios servicios, consulte el repositorio de ejemplos GitHub del SDK de AWS documentación.

Uso de Amazon Rekognition y Lambda para etiquetar activos en un bucket de Amazon S3

En este tutorial, creará una AWS Lambda función que etiquete automáticamente los activos digitales ubicados en un bucket de Amazon S3. La función de Lambda lee todos los objetos de

un bucket de Amazon S3 determinado. Para cada objeto del bucket, pasa la imagen al servicio Amazon Rekognition para generar una serie de etiquetas. Cada etiqueta se usa para crear una etiqueta que se aplica a la imagen. Tras ejecutar la función de Lambda, esta crea automáticamente etiquetas basadas en todas las imágenes de un determinado bucket de Amazon S3 y las aplica a las imágenes.

Por ejemplo, supongamos que ejecuta la función de Lambda y tiene esta imagen en un bucket de Amazon S3.



A continuación, la aplicación crea etiquetas automáticamente y las aplica a la imagen.

Tags (6)

Track storage cost of other criteria by tagging your objects. **Learn more**

Key	Value
Nature	99.99188
Volcano	97.60948
Eruption	96.54574
Lava	79.63064
Mountain	99.99188
Outdoors	99.99188

> **Note**
>
> Los servicios que utiliza en este tutorial forman parte de la capa AWS gratuita. Cuando termine con el tutorial, le recomendamos que termine todos los recursos que haya creado durante el tutorial para que no se le cobre nada.

En este tutorial se utiliza la versión 2 del AWS SDK for Java. Consulte el GitHub repositorio de ejemplos del SDK de AWS documentación para ver tutoriales adicionales sobre Java V2.

Temas

- Requisitos previos
- Configuración del rol de IAM para Lambda
- Creación del proyecto
- Escribir el código
- Empaquetar el proyecto
- Implementación de la función de Lambda
- Probar el método Lambda

Requisitos previos

Antes de empezar, debe completar los pasos de Configuración del AWS SDK para Java. Después, asegúrese de cuenta con lo siguiente:

- Java 1.8 JDK.
- Maven 3.6 o superior.
- Un bucket de Amazon S3 con entre 5 y 7 imágenes de la naturaleza. La función de Lambda lee estas imágenes.

Configuración del rol de IAM para Lambda

En este tutorial se utilizan los servicios Amazon Rekognition y Amazon S3. Configure la función lambda-support para que tenga políticas que le permitan invocar estos servicios desde una función de Lambda.

Para configurar el rol

1. Inicie sesión en la consola de IAM Consola de administración de AWS y ábrala en https://console.aws.amazon.com/iam/.
2. En el panel de navegación, elija Roles y, a continuación, Crear rol.
3. Elija Servicio de AWS y, a continuación, Lambda.
4. Elija la pestaña Permisos.

5. Busque la opción AWSLambdaBasicExecutionRole.
6. Elija Siguiente: etiquetas.
7. Elija Revisar.
8. Asigne el nombre lambda-support al rol.
9. Elija Crear rol.
10. Seleccione lambda-support para ver la página de información general.
11. Seleccione Asociar políticas.
12. Elija una AmazonRekognitionFullAccessde las políticas de la lista.
13. Elija Asociar política.
14. Busque AmazonS3 yFullAccess, a continuación, seleccione Adjuntar política.

Creación del proyecto

Cree un nuevo proyecto de Java y, a continuación, configure el archivo pom.xml de Maven con los ajustes y dependencias necesarios. Asegúrese de que el archivo pom.xml tiene el siguiente aspecto:

```
<?xml version="1.0" encoding="UTF-8"?>
 <project xmlns="http://maven.apache.org/POM/4.0.0"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/
maven-4.0.0.xsd">
<modelVersion>4.0.0</modelVersion>
<groupId>org.example</groupId>
<artifactId>WorkflowTagAssets</artifactId>
<version>1.0-SNAPSHOT</version>
<packaging>jar</packaging>
<name>java-basic-function</name>
<properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    <maven.compiler.source>1.8</maven.compiler.source>
    <maven.compiler.target>1.8</maven.compiler.target>
</properties>
<dependencyManagement>
    <dependencies>
        <dependency>
            <groupId>software.amazon.awssdk</groupId>
            <artifactId>bom</artifactId>
            <version>2.10.54</version>
```

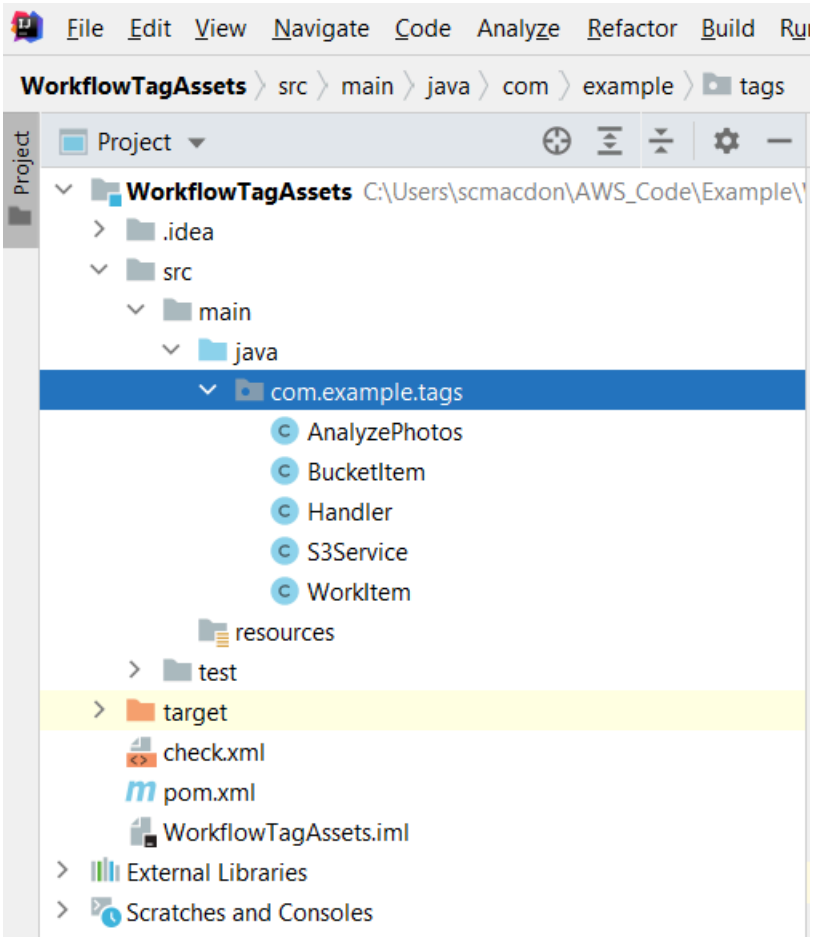
```
            <type>pom</type>
            <scope>import</scope>
        </dependency>
    </dependencies>
</dependencyManagement>
<dependencies>
   <dependency>
        <groupId>com.amazonaws</groupId>
        <artifactId>aws-lambda-java-core</artifactId>
        <version>1.2.1</version>
    </dependency>
    <dependency>
        <groupId>com.google.code.gson</groupId>
        <artifactId>gson</artifactId>
        <version>2.8.6</version>
    </dependency>
    <dependency>
        <groupId>org.apache.logging.log4j</groupId>
        <artifactId>log4j-api</artifactId>
        <version>2.10.0</version>
    </dependency>
    <dependency>
        <groupId>org.apache.logging.log4j</groupId>
        <artifactId>log4j-core</artifactId>
        <version>2.13.0</version>
        <scope>test</scope>
    </dependency>
    <dependency>
        <groupId>org.apache.logging.log4j</groupId>
        <artifactId>log4j-slf4j18-impl</artifactId>
        <version>2.13.3</version>
        <scope>test</scope>
    </dependency>
    <dependency>
        <groupId>org.junit.jupiter</groupId>
        <artifactId>junit-jupiter-api</artifactId>
        <version>5.6.0</version>
        <scope>test</scope>
    </dependency>
    <dependency>
        <groupId>org.junit.jupiter</groupId>
        <artifactId>junit-jupiter-engine</artifactId>
        <version>5.6.0</version>
        <scope>test</scope>
```

```
    </dependency>
    <dependency>
        <groupId>com.googlecode.json-simple</groupId>
        <artifactId>json-simple</artifactId>
        <version>1.1.1</version>
    </dependency>
    <dependency>
        <groupId>software.amazon.awssdk</groupId>
        <artifactId>s3</artifactId>
    </dependency>
    <dependency>
        <groupId>software.amazon.awssdk</groupId>
        <artifactId>rekognition</artifactId>
    </dependency>
</dependencies>
<build>
    <plugins>
        <plugin>
            <artifactId>maven-surefire-plugin</artifactId>
            <version>2.22.2</version>
        </plugin>
        <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-shade-plugin</artifactId>
            <version>3.2.2</version>
            <configuration>
                <createDependencyReducedPom>false</createDependencyReducedPom>
            </configuration>
            <executions>
                <execution>
                    <phase>package</phase>
                    <goals>
                        <goal>shade</goal>
                    </goals>
                </execution>
            </executions>
        </plugin>
        <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-compiler-plugin</artifactId>
            <version>3.8.1</version>
            <configuration>
                <source>1.8</source>
                <target>1.8</target>
```

```
            </configuration>
        </plugin>
    </plugins>
  </build>
</project>
```

Escribir el código

Utilice la API AWS Lambda de Java en tiempo de ejecución para crear la clase Java que define la función Lambda. En este ejemplo, hay una clase de Java para la función de Lambda denominada Handler y se requieren clases adicionales para este caso de uso. En el siguiente gráfico, se muestran las clases de Java en el proyecto. Observe que todas las clases de Java se encuentran en un paquete denominado com.example.tags.



Cree las siguientes clases de Java para el código:

- Handler usa la API de tiempo de ejecución Lambda Java y lleva a cabo el caso de uso descrito en este tutorial. AWS La lógica de la aplicación que se ejecuta se encuentra en el método handleRequest.
- S3Service usa la API de Amazon S3 para realizar operaciones de S3.
- AnalyzePhotosutiliza la API Amazon Rekognition para analizar las imágenes.
- BucketItemdefine un modelo que almacena la información del bucket de Amazon S3.
- WorkItemdefine un modelo que almacena los datos de Amazon Rekognition.

Clase Handler

Este código Java representa la clase Handler. La clase lee un indicador que se pasa a la función de Lambda. El servicio S3. ListBucketObjectsdevuelve un objeto List donde cada elemento es un valor de cadena que representa la clave del objeto. Si el valor del indicador es verdadero, las etiquetas se aplican iterando por la lista y aplicándolas a cada objeto mediante una llamada al método s3Service.tagAssets. Si el valor del indicador es falso, entonces el S3Service. deleteTagFromSe invoca el método Object que elimina las etiquetas. Además, ten en cuenta que puedes registrar mensajes en CloudWatch los registros de Amazon mediante un LambdaLoggerobjeto.

> **Note**
>
> Asegúrese de asignar el nombre de su bucket a la variable bucketName.

```
package com.example.tags;

import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.LambdaLogger;
import java.util.ArrayList;
import java.util.List;
import java.util.Map;

public class Handler implements RequestHandler<Map<String,String>, String> {

@Override
public String handleRequest(Map<String, String> event, Context context) {
    LambdaLogger logger = context.getLogger();
    String delFlag = event.get("flag");
```

```
    logger.log("FLAG IS: " + delFlag);
    S3Service s3Service = new S3Service();
    AnalyzePhotos photos = new AnalyzePhotos();

    String bucketName = "<Enter your bucket name>";
    List<String> myKeys = s3Service.listBucketObjects(bucketName);
    if (delFlag.compareTo("true") == 0) {

        // Create a List to store the data.
        List<ArrayList<WorkItem>> myList = new ArrayList<>();

        // loop through each element in the List and tag the assets.
        for (String key : myKeys) {

            byte[] keyData = s3Service.getObjectBytes(bucketName, key);

            // Analyze the photo and return a list where each element is a WorkItem.
            ArrayList<WorkItem> item = photos.detectLabels(keyData, key);
            myList.add(item);
        }

        s3Service.tagAssets(myList, bucketName);
        logger.log("All Assets in the bucket are tagged!");

    } else {

        // Delete all object tags.
        for (String key : myKeys) {
            s3Service.deleteTagFromObject(bucketName, key);
            logger.log("All Assets in the bucket are deleted!");
        }
     }
    return delFlag;
 }
}
```

Clase de servicio de S3

La siguiente clase usa la API de Amazon S3 para realizar operaciones de S3. Por ejemplo, el getObjectBytesmétodo devuelve una matriz de bytes que representa la imagen. Del mismo modo, el listBucketObjectsmétodo devuelve un objeto List en el que cada elemento es un valor de cadena que especifica el nombre de la clave.

```
package com.example.tags;

import software.amazon.awssdk.core.ResponseBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.PutObjectTaggingRequest;
import software.amazon.awssdk.services.s3.model.GetObjectResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.ListObjectsResponse;
import software.amazon.awssdk.services.s3.model.S3Object;
import software.amazon.awssdk.services.s3.model.GetObjectTaggingResponse;
import software.amazon.awssdk.services.s3.model.ListObjectsRequest;
import java.util.ArrayList;
import java.util.List;
import software.amazon.awssdk.services.s3.model.Tagging;
import software.amazon.awssdk.services.s3.model.Tag;
import software.amazon.awssdk.services.s3.model.GetObjectTaggingRequest;
import software.amazon.awssdk.services.s3.model.DeleteObjectTaggingRequest;

public class S3Service {

private S3Client getClient() {

   Region region = Region.US_WEST_2;
   return S3Client.builder()
           .region(region)
           .build();
}

public byte[] getObjectBytes(String bucketName, String keyName) {

   S3Client s3 = getClient();

   try {

       GetObjectRequest objectRequest = GetObjectRequest
               .builder()
               .key(keyName)
               .bucket(bucketName)
               .build();

       // Return the byte[] from this object.
```

```
        ResponseBytes<GetObjectResponse> objectBytes =
s3.getObjectAsBytes(objectRequest);
        return objectBytes.asByteArray();

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return null;
}

// Returns the names of all images in the given bucket.
public List<String> listBucketObjects(String bucketName) {

    S3Client s3 = getClient();
    String keyName;

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

    try {
        ListObjectsRequest listObjects = ListObjectsRequest
                .builder()
                .bucket(bucketName)
                .build();

        ListObjectsResponse res = s3.listObjects(listObjects);
        List<S3Object> objects = res.contents();

        for (S3Object myValue: objects) {
            keyName = myValue.key();
            keys.add(keyName);
        }
        return keys;

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return null;
}

// Tag assets with labels in the given list.
public void tagAssets(List myList, String bucketName) {
```

```
    try {

        S3Client s3 = getClient();
        int len = myList.size();

        String assetName = "";
        String labelName = "";
        String labelValue = "";

        // Tag all the assets in the list.
        for (Object o : myList) {

            // Need to get the WorkItem from each list.
            List innerList = (List) o;
            for (Object value : innerList) {

                WorkItem workItem = (WorkItem) value;
                assetName = workItem.getKey();
                labelName = workItem.getName();
                labelValue = workItem.getConfidence();
                tagExistingObject(s3, bucketName, assetName, labelName, labelValue);
            }
        }

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

// This method tags an existing object.
private void tagExistingObject(S3Client s3, String bucketName, String key, String
label, String LabelValue) {

    try {

        // First need to get existing tag set; otherwise the existing tags are
overwritten.
        GetObjectTaggingRequest getObjectTaggingRequest =
GetObjectTaggingRequest.builder()
                .bucket(bucketName)
                .key(key)
                .build();
```

```
        GetObjectTaggingResponse response =
s3.getObjectTagging(getObjectTaggingRequest);

        // Get the existing immutable list - cannot modify this list.
        List<Tag> existingList = response.tagSet();
        ArrayList<Tag> newTagList = new ArrayList(new ArrayList<>(existingList));

        // Create a new tag.
        Tag myTag = Tag.builder()
                .key(label)
                .value(LabelValue)
                .build();

        // push new tag to list.
        newTagList.add(myTag);
        Tagging tagging = Tagging.builder()
                .tagSet(newTagList)
                .build();

        PutObjectTaggingRequest taggingRequest = PutObjectTaggingRequest.builder()
                .key(key)
                .bucket(bucketName)
                .tagging(tagging)
                .build();

        s3.putObjectTagging(taggingRequest);
        System.out.println(key + " was tagged with " + label);

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
 }

// Delete tags from the given object.
public void deleteTagFromObject(String bucketName, String key) {

    try {

        DeleteObjectTaggingRequest deleteObjectTaggingRequest =
DeleteObjectTaggingRequest.builder()
                .key(key)
                .bucket(bucketName)
                .build();
```

```
        S3Client s3 = getClient();
        s3.deleteObjectTagging(deleteObjectTaggingRequest);

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
  }
}
```

AnalyzePhotos clase

El siguiente código Java representa la AnalyzePhotosclase. Esta clase utiliza la API de Amazon Rekognition para analizar las imágenes.

```
package com.example.tags;

import software.amazon.awssdk.auth.credentials.EnvironmentVariableCredentialsProvider;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.Image;
import software.amazon.awssdk.services.rekognition.model.DetectLabelsRequest;
import software.amazon.awssdk.services.rekognition.model.DetectLabelsResponse;
import software.amazon.awssdk.services.rekognition.model.Label;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import java.util.ArrayList;
import java.util.List;

public class AnalyzePhotos {

 // Returns a list of WorkItem objects that contains labels.
 public ArrayList<WorkItem> detectLabels(byte[] bytes, String key) {

    Region region = Region.US_EAST_2;
    RekognitionClient rekClient = RekognitionClient.builder()
            .credentialsProvider(EnvironmentVariableCredentialsProvider.create())
            .region(region)
            .build();

    try {
```

```
        SdkBytes sourceBytes = SdkBytes.fromByteArray(bytes);

        // Create an Image object for the source image.
        Image souImage = Image.builder()
                .bytes(sourceBytes)
                .build();

        DetectLabelsRequest detectLabelsRequest = DetectLabelsRequest.builder()
                .image(souImage)
                .maxLabels(10)
                .build();

        DetectLabelsResponse labelsResponse =
 rekClient.detectLabels(detectLabelsRequest);

        // Write the results to a WorkItem instance.
        List<Label> labels = labelsResponse.labels();
        ArrayList<WorkItem> list = new ArrayList<>();
        WorkItem item ;
        for (Label label: labels) {
            item = new WorkItem();
            item.setKey(key); // identifies the photo.
            item.setConfidence(label.confidence().toString());
            item.setName(label.name());
            list.add(item);
        }
        return list;

    } catch (RekognitionException e) {
        System.out.println(e.getMessage());
        System.exit(1);
    }
    return null ;
  }
}
```

BucketItem clase

El siguiente código Java representa la BucketItemclase que almacena los datos de objetos de Amazon S3.

```
package com.example.tags;
```

```
public class BucketItem {

 private String key;
 private String owner;
 private String date ;
 private String size ;


 public void setSize(String size) {
    this.size = size ;
 }

 public String getSize() {
    return this.size ;
 }

 public void setDate(String date) {
    this.date = date ;
 }

 public String getDate() {
    return this.date ;
 }

 public void setOwner(String owner) {
    this.owner = owner ;
 }

 public String getOwner() {
    return this.owner ;
 }

 public void setKey(String key) {
    this.key = key ;
 }

 public String getKey() {
    return this.key ;
 }
}
```

WorkItem clase

El siguiente código Java representa la WorkItemclase.

```
 package com.example.tags;

 public class WorkItem {

 private String key;
 private String name;
 private String confidence ;

public void setKey (String key) {
    this.key = key;
}

public String getKey() {
    return this.key;
}

public void setName (String name) {
    this.name = name;
}

public String getName() {
    return this.name;
}

public void setConfidence (String confidence) {
    this.confidence = confidence;
}

public String getConfidence() {
    return this.confidence;
}

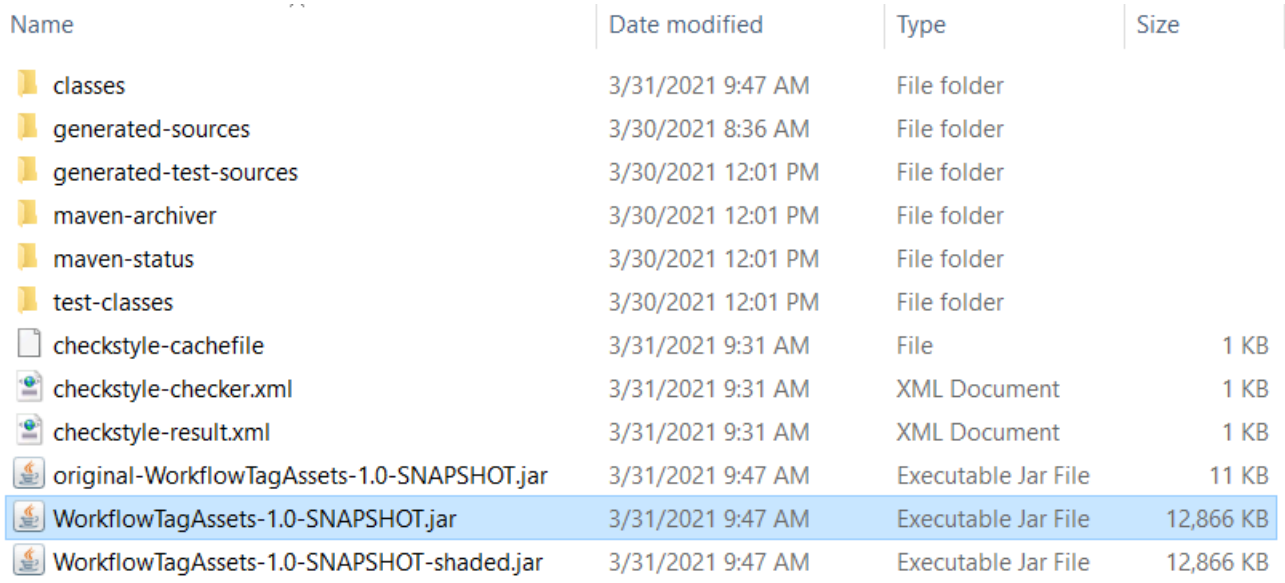
}
```

Empaquetar el proyecto

Empaquete el proyecto en un archivo.jar (JAR) mediante el siguiente comando de Maven.

```
mvn package
```

El archivo JAR se encuentra en la carpeta de destino (que es una carpeta secundaria de la carpeta del proyecto).



Name	Date modified	Type	Size
classes	3/31/2021 9:47 AM	File folder	
generated-sources	3/30/2021 8:36 AM	File folder	
generated-test-sources	3/30/2021 12:01 PM	File folder	
maven-archiver	3/30/2021 12:01 PM	File folder	
maven-status	3/30/2021 12:01 PM	File folder	
test-classes	3/30/2021 12:01 PM	File folder	
checkstyle-cachefile	3/31/2021 9:31 AM	File	1 KB
checkstyle-checker.xml	3/31/2021 9:31 AM	XML Document	1 KB
checkstyle-result.xml	3/31/2021 9:31 AM	XML Document	1 KB
original-WorkflowTagAssets-1.0-SNAPSHOT.jar	3/31/2021 9:47 AM	Executable Jar File	11 KB
WorkflowTagAssets-1.0-SNAPSHOT.jar	3/31/2021 9:47 AM	Executable Jar File	12,866 KB
WorkflowTagAssets-1.0-SNAPSHOT-shaded.jar	3/31/2021 9:47 AM	Executable Jar File	12,866 KB

> **Note**
>
> Observe el uso de maven-shade-pluginen el archivo POM del proyecto. Este complemento es responsable de crear un JAR que contenga las dependencias necesarias. Si intentas empaquetar el proyecto sin este complemento, las dependencias necesarias no se incluyen en el archivo JAR y encontrarás un ClassNotFoundException.

Implementación de la función de Lambda

1. Abra la consola de Lambda.
2. Elija Crear función.
3. Elija Crear desde cero.
4. En la sección Información básica, introduzca cron como nombre.
5. En Tiempo de ejecución, elija Java 8.
6. Elija Usar un rol existente y, a continuación, elija lambda-support (el rol de IAM que creó).
7. Seleccione Creación de función.
8. Para Tipo de entrada de código, elija Cargar un archivo .zip.

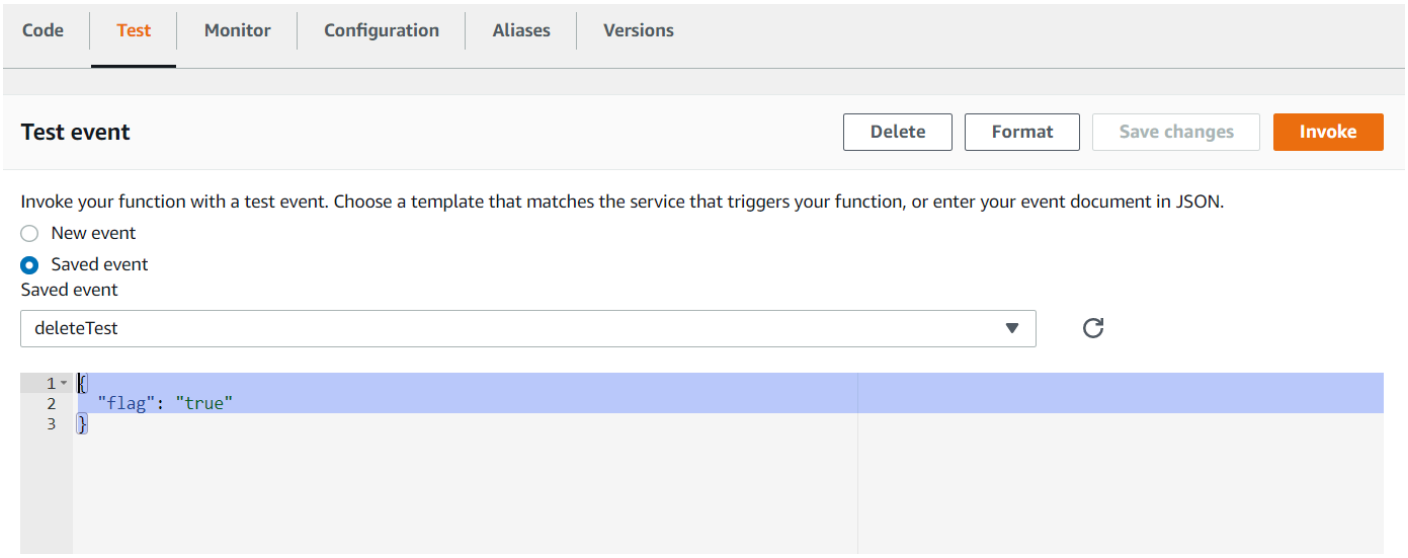
9. Seleccione Cargar y, a continuación, busque el archivo JAR que creó.
10. Para Handler, introduzca el nombre completo de la función, por ejemplo, com.example.tags.Handler:handleRequest (com.example.tags especifica el paquete, Handler es la clase seguida de :: y el nombre del método).
11. Seleccione Save.

Probar el método Lambda

En este punto del tutorial, puede probar la función de Lambda.

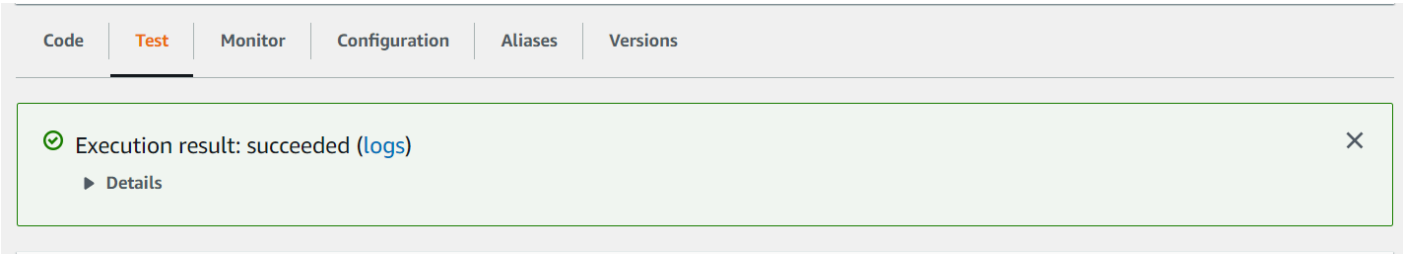
1. En la consola de Lambda, haga clic en la pestaña Prueba y, a continuación, introduzca el siguiente JSON.

```
                {
"flag": "true"
 }
```



> **Note**
>
> Al transferir true se etiquetan los activos digitales y al transferir false se eliminan las etiquetas.

2. Pulse el botón Invocar. Tras invocar la función de Lambda, verá un mensaje de éxito.



Enhorabuena, ha creado una AWS Lambda función que aplica etiquetas automáticamente a los activos digitales ubicados en un bucket de Amazon S3. Como se indicó al principio de este tutorial, recuerde cancelar todos los recursos que creó mientras realiza este tutorial para asegurarse de que no se le cobre nada.

Para ver más ejemplos de AWS varios servicios, consulte el repositorio de ejemplos del SDK de AWSdocumentación. GitHub

Creación de aplicaciones de análisis de vídeo de AWS

Puede crear una aplicación web Java que analice los vídeos para detectar etiquetas mediante el AWS SDK para Java versión 2. La aplicación creada en este tutorial de AWS le permite subir un vídeo (archivo MP4) a un bucket de Amazon S3. A continuación, la aplicación utiliza el servicio Amazon Rekognition para analizar el vídeo. Los resultados se utilizan para rellenar un modelo de datos y, a continuación, se genera un informe que se envía por correo electrónico a un usuario específico mediante el servicio de correo electrónico Amazon Simple.

La siguiente ilustración muestra un informe que se genera después de que la aplicación termine de analizar el vídeo. Las columnas de la tabla siguiente muestran los valores de rango de edad, barba, gafas y ojos abiertos, así como valores de confianza para distintas predicciones de atributos.

	Age Range	Beard	Eye glasses	Eyes open
2				
3	AgeRange(Low=38, High=56)	Beard(Value=false, Confidence=83.07253)	Eyeglasses(Value=true, Confidence=55.965977)	EyeOpen(Value=true, Confidence=94.691696)
4	AgeRange(Low=36, High=52)	Beard(Value=true, Confidence=50.721912)	Eyeglasses(Value=false, Confidence=63.886036)	EyeOpen(Value=true, Confidence=95.906364)
5	AgeRange(Low=51, High=69)	Beard(Value=true, Confidence=58.38352)	Eyeglasses(Value=false, Confidence=96.39576)	EyeOpen(Value=true, Confidence=53.580643)
6	AgeRange(Low=49, High=67)	Beard(Value=false, Confidence=81.41662)	Eyeglasses(Value=true, Confidence=65.28722)	EyeOpen(Value=true, Confidence=95.11523)
7	AgeRange(Low=51, High=69)	Beard(Value=true, Confidence=61.533833)	Eyeglasses(Value=false, Confidence=97.51163)	EyeOpen(Value=true, Confidence=82.21834)
8	AgeRange(Low=29, High=45)	Beard(Value=false, Confidence=74.22591)	Eyeglasses(Value=true, Confidence=64.906685)	EyeOpen(Value=true, Confidence=98.48175)
9	AgeRange(Low=51, High=69)	Beard(Value=true, Confidence=65.9394)	Eyeglasses(Value=false, Confidence=94.14824)	EyeOpen(Value=true, Confidence=94.857346)
10	AgeRange(Low=44, High=62)	Beard(Value=true, Confidence=78.648)	Eyeglasses(Value=true, Confidence=65.83134)	EyeOpen(Value=true, Confidence=98.538666)
11				

En este tutorial, creará una aplicación Spring Boot que invoque varios servicios de AWS. Las API de Spring Boot se utilizan para crear un modelo, diferentes vistas y un controlador. Para obtener más información, consulte Spring Boot.

Este servicio utiliza los siguientes servicios de AWS:

- Amazon Rekognition
- Amazon S3
- Amazon SES
- AWS Elastic Beanstalk

Los servicios de AWS incluidos en este tutorial están incluidos en la Capa gratuita de AWS. Le recomendamos que finalice todos los recursos que haya creado en el tutorial cuando haya terminado de usarlos para evitar que se le cobre por ellos.

Requisitos previos

Antes de empezar, debe completar los pasos de configuración del AWS SDK para Java. Después, asegúrese de cuenta con lo siguiente:

- Java 1.8 JDK.
- Maven 3.6 o posterior.
- Un bucket de Amazon S3 llamado vídeo[algún-valor]. Asegúrese de usar este nombre de bucket en el código Java de Amazon S3. Para obtener más información, consulte Creación de un bucket.
- Un rol de IAM. Lo necesitará para la clase VideoDetectFaces que va a crear. Para obtener más información, consulte Configuring Amazon Rekognition Video.
- Un tema válido de Amazon SNS. Lo necesitará para la clase VideoDetectFaces que va a crear. Para obtener más información, consulte Configuring Amazon Rekognition Video.

Procedimiento

En el transcurso del tutorial, aprenderá a hacer lo siguiente:

1. Crear un proyecto
2. Agregar las dependencias de POM a su proyecto
3. Crear las clases de Java
4. Crear los archivos HTML
5. Crear los archivos de script
6. Empaquetar el proyecto en un archivo JAR
7. Implementar la aplicación en AWS Elastic Beanstalk

Para continuar con el tutorial, siga las instrucciones detalladas del repositorio de ejemplos de AWS SDK en la documentación en GitHub.

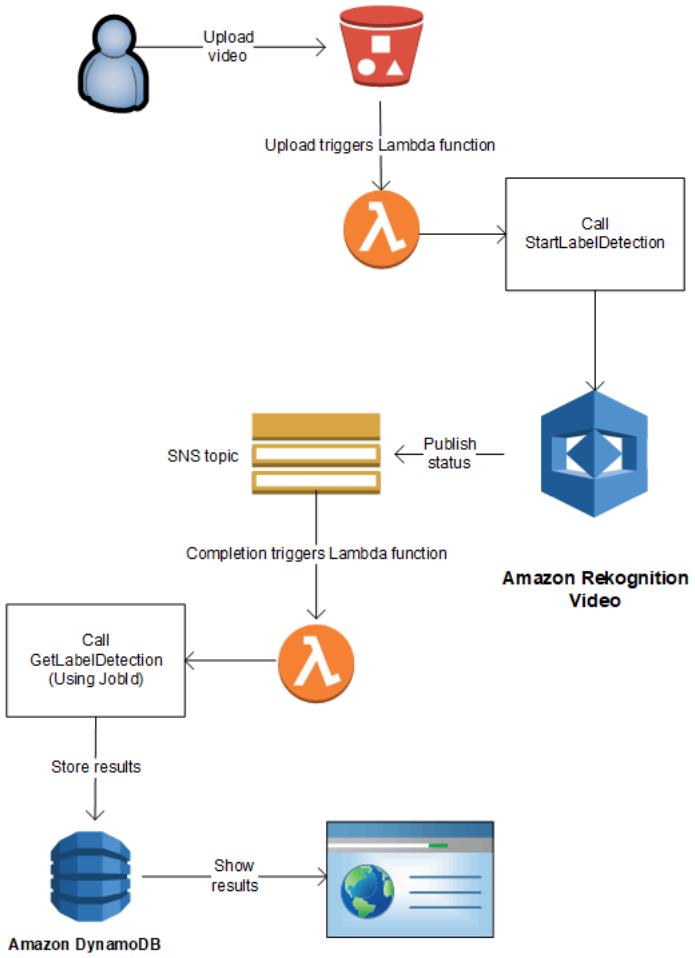
Creación de una función de Lambda de Amazon Rekognition

En este tutorial, se muestra cómo obtener los resultados de una operación de análisis de vídeo para la detección de etiquetas mediante una función de Lambda en Java.

> **Note**
>
> En este tutorial se utiliza el AWS SDK for Java 1.x. Para ver un tutorial sobre el uso de Rekognition y la versión 2 del SDK para AWS Java, consulta AWS el repositorio de ejemplos del SDK de documentación. GitHub

Puede utilizar las funciones de Lambda con las operaciones de Amazon Rekognition Video. Por ejemplo, el siguiente diagrama muestra un sitio web que utiliza una función de Lambda para comenzar automáticamente el análisis de un vídeo cuando este se carga en un bucket de Amazon S3. Cuando se activa la función Lambda, llama StartLabelDetectionpara empezar a detectar etiquetas en el vídeo cargado. Para obtener información sobre cómo utilizar Lambda para procesar notificaciones de eventos desde un bucket de Amazon S3, consulte Uso de AWS Lambda con eventos de Amazon S3.

Una segunda función de Lambda se activa cuando se envía el estado de realización del análisis al tema de Amazon SNS registrado. La segunda función Lambda llama GetLabelDetectionpara obtener los resultados del análisis. A continuación, los resultados se almacenan en una base de datos como paso previo antes de mostrarlos en una página web. Esta segunda función lambda es el objetivo de este tutorial.



En este tutorial, la función de Lambda se activa cuando Amazon Rekognition Video envía el estado de realización del análisis de vídeo al tema de Amazon SNS registrado. A continuación, la función recopila los resultados del análisis de vídeo llamando a GetLabelDetection. Con fines de demostración, este tutorial escribe los resultados de la detección de etiquetas en un CloudWatch registro. La función de Lambda de la aplicación debería almacenar los resultados del análisis para su uso posterior. Por ejemplo, puede utilizar Amazon DynamoDB para guardar los resultados del análisis. Para obtener más información, consulte el tema relacionado con el uso de DynamoDB.

Los siguientes procedimientos le muestran cómo:

- Crear el tema de Amazon SNS y configurar los permisos.
- Cree la función Lambda mediante el tema Amazon SNS Consola de administración de AWS y suscríbala al tema.
- Configurar la función de Lambda mediante la Consola de administración de AWS.
- Agregue código de muestra a un AWS Toolkit for Eclipse proyecto y cárguelo en la función Lambda.

- Probar la función de Lambda mediante la AWS CLI.

> **Note**
>
> Utilice la misma AWS región en todo el tutorial.

Requisitos previos

En este tutorial, se supone que está familiarizado con AWS Toolkit for Eclipse. Para obtener más información, consulte AWS Toolkit for Eclipse.

Creación del tema de SNS

El estado de realización de una operación de análisis de vídeo de Amazon Rekognition Video se envía a un tema de Amazon SNS. Este procedimiento crea el tema de Amazon SNS y el rol de servicio de IAM que otorga a Amazon Rekognition Video acceso a los temas de Amazon SNS. Para obtener más información, consulte Cómo llamar a las operaciones de Amazon Rekognition Video.

Para crear un tema de Amazon SNS

1. Si no lo ha hecho aún, cree un rol de servicio de IAM para otorgar a Amazon Rekognition Video acceso a sus temas de Amazon SNS. Anote el nombre de recurso de Amazon (ARN). Para obtener más información, consulte Otorgar acceso a varios temas de Amazon SNS.
2. Cree un tema de Amazon SNS mediante la consola de Amazon SNS. Sólo tiene que especificar el nombre del tema. Añada al nombre del tema un prefijo. AmazonRekognition Apunte el ARN del tema.

Crear la función de Lambda

Utilice la función de Lambda para crear la función de Consola de administración de AWS. A continuación, deberá usar un proyecto de AWS Toolkit for Eclipse para subir el paquete de la función de Lambda en AWS Lambda. También es posible crear la función de Lambda con AWS Toolkit for Eclipse. Para obtener más información, consulte Tutorial: Cómo crear, cargar e invocar una función de AWS Lambda.

Para crear la función de Lambda

1. Inicie sesión en la consola de administración de AWS y abra la consola de AWS Lambda en https://console.aws.amazon.com/lambda/.
2. Elija Crear función.
3. Elija Crear desde cero.
4. En Nombre de función, introduzca un nombre para la función.
5. En Tiempo de ejecución, elija Java 8.
6. Elija Seleccionar o crear un rol de ejecución.
7. En Rol de ejecución, elija Crear un nuevo rol con permisos básicos de Lambda.
8. Observe el nombre del nuevo rol que se muestra en la parte inferior de la sección Información básica.
9. Seleccione Creación de función.

Configuración de la función de Lambda

Una vez que haya creado la función de Lambda, puede configurarla para que la active el tema de Amazon SNS que se crea en Creación del tema de SNS. También deberá ajustar los requisitos de memoria y el periodo de tiempo de espera para la función de Lambda.

Para configurar la función de Lambda

1. En Código de función, escriba `com.amazonaws.lambda.demo.JobCompletionHandler` para Controlador.
2. En Configuración básica, elija Editar. Se muestra el cuadro de diálogo Editar configuración básica.
   a. Elija 1024 para Memoria.
   b. Elija 10 segundos para Tiempo de espera.
   c. Seleccione Save.
3. En Diseñador, elija + Agregar desencadenador. Se muestra el cuadro de diálogo Agregar desencadenador.
4. En Configuración de desencadenador elija SNS.

   En Tema de SNS, elija el tema de Amazon SNS que creó en Creación del tema de SNS.

5. Elija Activar disparador.
6. Para añadir el disparador, elija Añadir.
7. Elija Guardar para guardar la función de Lambda actualizada.

Configuración del rol de IAM para Lambda

Para llamar a Amazon Rekognition Video operations, debe añadir la política gestionada de AmazonRekognitionFullAccessAWS a la función Lambda de IAM. Las operaciones de inicio, por ejemplo StartLabelDetection, también requieren permisos de pase para el rol de servicio de IAM que Amazon Rekognition Video utiliza para acceder al tema de Amazon SNS.

Para configurar el rol

1. Inicie sesión en la consola de IAM Consola de administración de AWS y ábrala en. https://console.aws.amazon.com/iam/
2. Seleccione Roles en el panel de navegación.
3. En la lista, elija el nombre del rol de ejecución que ha creado en Crear la función de Lambda.
4. Elija la pestaña Permisos.
5. Seleccione Asociar políticas.
6. Elija una AmazonRekognitionFullAccessde las políticas de la lista.
7. Elija Asociar política.
8. Nuevamente, elija el rol de ejecución.
9. Elija Agregar política insertada.
10. Seleccione la pestaña JSON.
11. Reemplace la política existente por la siguiente política. Reemplace `servicerole` por el rol de servicio de IAM; que ha creado en Creación del tema de SNS.
12. Elija Revisar política.
13. En Nombre*, escriba un nombre para la política.
14. Elija Crear política.

Crear el proyecto AWS Toolkit for Eclipse Lambda

Cuando se activa la función Lambda, el código siguiente obtiene el estado de finalización del tema Amazon SNS y GetLabelDetectionllama para obtener los resultados del análisis. Se graba en un

registro un recuento de las etiquetas detectadas y una lista de las etiquetas detectadas. CloudWatch La función de Lambda debería almacenar los resultados del análisis de vídeo para su uso posterior.

Para crear el proyecto AWS Toolkit for Eclipse Lambda

1. Cree un proyecto AWS Toolkit for EclipseAWS Lambda.
   - En Nombre del proyecto, escriba un nombre de proyecto de su elección.
   - Para el nombre de la clase:, introduzca JobCompletionHandler.
   - En Tipo de entrada, elija Evento de SNS.
   - Deje el resto de los campos sin modificar.
2. En el explorador de proyectos de Eclipse, abra el método de controlador de Lambda generado (JobCompletionHandler.java) y sustituya el contenido por lo siguiente:

```
//Copyright 2018 Amazon.com, Inc. or its affiliates. All Rights Reserved.
//PDX-License-Identifier: MIT-0 (For details, see https://github.com/awsdocs/
amazon-rekognition-developer-guide/blob/master/LICENSE-SAMPLECODE.)

package com.amazonaws.lambda.demo;

import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.LambdaLogger;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.SNSEvent;
import java.util.List;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.rekognition.AmazonRekognition;
import com.amazonaws.services.rekognition.AmazonRekognitionClientBuilder;
import com.amazonaws.services.rekognition.model.GetLabelDetectionRequest;
import com.amazonaws.services.rekognition.model.GetLabelDetectionResult;
import com.amazonaws.services.rekognition.model.LabelDetection;
import com.amazonaws.services.rekognition.model.LabelDetectionSortBy;
import com.amazonaws.services.rekognition.model.VideoMetadata;
import com.fasterxml.jackson.databind.JsonNode;
import com.fasterxml.jackson.databind.ObjectMapper;



public class JobCompletionHandler implements RequestHandler<SNSEvent, String> {

   @Override
```

```
public String handleRequest(SNSEvent event, Context context) {

   String message = event.getRecords().get(0).getSNS().getMessage();
   LambdaLogger logger = context.getLogger();

   // Parse SNS event for analysis results. Log results
   try {
      ObjectMapper operationResultMapper = new ObjectMapper();
      JsonNode jsonResultTree = operationResultMapper.readTree(message);
      logger.log("Rekognition Video Operation:========================");
      logger.log("Job id: " + jsonResultTree.get("JobId"));
      logger.log("Status : " + jsonResultTree.get("Status"));
      logger.log("Job tag : " + jsonResultTree.get("JobTag"));
      logger.log("Operation : " + jsonResultTree.get("API"));

      if (jsonResultTree.get("API").asText().equals("StartLabelDetection")) {

         if (jsonResultTree.get("Status").asText().equals("SUCCEEDED")){
            GetResultsLabels(jsonResultTree.get("JobId").asText(), context);
         }
         else{
            String errorMessage = "Video analysis failed for job "
                  + jsonResultTree.get("JobId")
                  + "State " + jsonResultTree.get("Status");
            throw new Exception(errorMessage);
         }

      } else
         logger.log("Operation not StartLabelDetection");

   } catch (Exception e) {
      logger.log("Error: " + e.getMessage());
      throw new RuntimeException (e);


   }

   return message;
}

void GetResultsLabels(String startJobId, Context context) throws Exception {

   LambdaLogger logger = context.getLogger();
```

```
     AmazonRekognition rek =
AmazonRekognitionClientBuilder.standard().withRegion(Regions.US_EAST_1).build();

     int maxResults = 1000;
     String paginationToken = null;
     GetLabelDetectionResult labelDetectionResult = null;
     String labels = "";
     Integer labelsCount = 0;
     String label = "";
     String currentLabel = "";

     //Get label detection results and log them.
     do {

        GetLabelDetectionRequest labelDetectionRequest = new
GetLabelDetectionRequest().withJobId(startJobId)

.withSortBy(LabelDetectionSortBy.NAME).withMaxResults(maxResults).withNextToken(pagination

        labelDetectionResult = rek.getLabelDetection(labelDetectionRequest);

        paginationToken = labelDetectionResult.getNextToken();
        VideoMetadata videoMetaData = labelDetectionResult.getVideoMetadata();

        // Add labels to log
        List<LabelDetection> detectedLabels = labelDetectionResult.getLabels();

        for (LabelDetection detectedLabel : detectedLabels) {
           label = detectedLabel.getLabel().getName();
           if (label.equals(currentLabel)) {
              continue;
           }
           labels = labels + label + " / ";
           currentLabel = label;
           labelsCount++;

        }
     } while (labelDetectionResult != null &&
labelDetectionResult.getNextToken() != null);

     logger.log("Total number of labels : " + labelsCount);
     logger.log("labels : " + labels);

  }
```

```
}
```

3. Los espacios de nombres de Rekognition no se resuelven. Para corregirlo:
   - Detenga el ratón sobre la parte subrayada de la línea `import com.amazonaws.services.rekognition.AmazonRekognition;`.
   - Elija Corregir configuración del proyecto....
   - Elija la versión más reciente del archivo de Amazon Rekognition.
   - Elija Aceptar para añadir el archivo al proyecto.
4. Guarde el archivo.
5. Haga clic con el botón derecho en la ventana de código de Eclipse, elija AWS Lambda y, a continuación, elija Cargar función a AWS Lambda.
6. En la página Seleccionar función de Lambda de destino, elija la región de AWS que desea utilizar.
7. Elija Elegir una función de Lambda existente y seleccione la función de Lambda que ha creado en Crear la función de Lambda.
8. Elija Siguiente. Se muestra el cuadro de diálogo Configuración de función.
9. En Rol de IAM, elija el rol de IAM que creó en Crear la función de Lambda.
10. Elija Finalizar y la función de Lambda se subirá en AWS.

Prueba de la función de Lambda

Utilice el siguiente AWS CLI comando para probar la función Lambda iniciando el análisis de detección de etiquetas de un vídeo. Una vez finalizado el análisis, la función de Lambda se activa. Compruebe que el análisis se ha realizado correctamente comprobando los CloudWatch registros.

Para probar la función de Lambda

1. Cargue un archivo de vídeo con formato MOV o MPEG-4 en el bucket de S3. Para realizar pruebas, cargue un vídeo con una duración inferior a 30 segundos.

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

2. Ejecute el siguiente AWS CLI comando para empezar a detectar etiquetas en un vídeo.

```
aws rekognition start-label-detection --video
 "S3Object={Bucket="bucketname",Name="videofile"}" \
--notification-channel "SNSTopicArn=TopicARN,RoleArn=RoleARN" \
--region Region
```

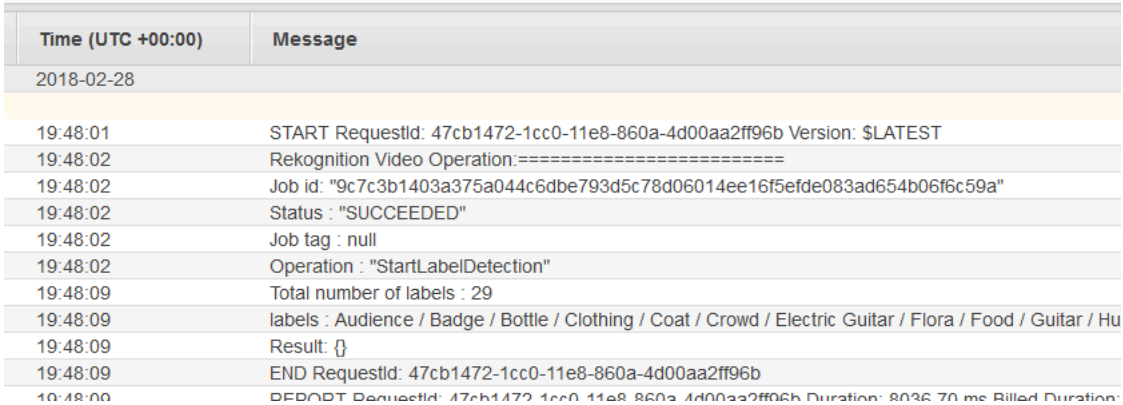
Actualice los siguientes valores:

- Cambie `bucketname` y `videofile` en el nombre de bucket de Amazon S3 y nombre de archivo del vídeo donde desea detectar etiquetas.
- Reemplace `TopicARN` por el ARN del tema de Amazon SNS que ha creado en Creación del tema de SNS.
- Reemplace `RoleARN` por el ARN del rol de IAM que ha creado en Creación del tema de SNS.
- Cambie `Region` a la AWS región que esté utilizando.

3. Anote el valor de `JobId` en la respuesta. La respuesta tiene un aspecto similar a la del siguiente ejemplo JSON.

```
{
    "JobId": "547089ce5b9a8a0e7831afa655f42e5d7b5c838553f1a584bf350ennnnnnnnnn"
}
```

4. Abra la consola de https://console.aws.amazon.com/cloudwatch/.
5. Cuando finalice el análisis, aparecerá una entrada de log para la función de Lambda en el Grupo de registro.
6. Elija la función de Lambda para ver los flujos de registro.
7. Elija el flujo de logs más reciente para ver las entradas de log realizadas por la función de Lambda. Si la operación se realizó correctamente, tendrá un aspecto similar al siguiente resultado, que muestra los detalles de la operación de reconocimiento de vídeo, incluido el identificador del trabajo, el tipo de operación StartLabelDetection "» y una lista de categorías de etiquetas detectadas, como Botella, Ropa, Multitud y Comida:



Time (UTC +00:00)	Message
2018-02-28	
19:48:01	START RequestId: 47cb1472-1cc0-11e8-860a-4d00aa2ff96b Version: $LATEST
19:48:02	Rekognition Video Operation:==========
19:48:02	Job id: "9c7c3b1403a375a044c6dbe793d5c78d06014ee16f5efde083ad654b06f6c59a"
19:48:02	Status : "SUCCEEDED"
19:48:02	Job tag : null
19:48:02	Operation : "StartLabelDetection"
19:48:09	Total number of labels : 29
19:48:09	labels : Audience / Badge / Bottle / Clothing / Coat / Crowd / Electric Guitar / Flora / Food / Guitar / Hu
19:48:09	Result: {}
19:48:09	END RequestId: 47cb1472-1cc0-11e8-860a-4d00aa2ff96b
19:48:09	REPORT RequestId: 47cb1472-1cc0-11e8-860a-4d00aa2ff96b Duration: 8036.70 ms Billed Duration:

El valor de ID de trabajo debería coincidir con el valor de `JobId` que anotó en el paso 3.

Uso de Amazon Rekognition para la verificación de identidad

Amazon Rekognition proporciona a los usuarios varias operaciones que permiten la creación sencilla de sistemas de verificación de identidad. Amazon Rekognition permite al usuario detectar rostros en una imagen y, a continuación, comparar los rostros detectados con otros mediante la comparación de los datos de los rostros. Estos datos faciales se almacenan en contenedores del lado del servidor denominados colecciones. Al utilizar las operaciones de detección de rostros, comparación de rostros y gestión de colecciones de Amazon Rekognition, puede crear una aplicación con una solución de verificación de identidad.

Este tutorial mostrará dos flujos de trabajo comunes para la creación de aplicaciones que requieren la verificación de identidad.

El primer flujo de trabajo implica el registro de un nuevo usuario en una colección. El segundo flujo de trabajo implica buscar en una colección existente con el fin de iniciar sesión como usuario habitual.

Usará el AWS SDK para Python en este tutorial. También puedes consultar el GitHub repositorio de ejemplos del SDK de AWS documentación para ver más tutoriales de Python.

Temas

- Requisitos previos
- Creación de una colección
- Nuevo registro de usuarios
- Inicio de sesión de usuario existente

Requisitos previos

Antes de comenzar este tutorial, necesitará instalar Python y completar los pasos necesarios para configurar el AWS SDK de Python. Además de esto, asegúrese de que:

- Ha creado una cuenta de AWS y un rol de IAM
- Ha instalado el SDK para Python (Boto3)
- Has configurado correctamente tus credenciales de AWS acceso
- Ha creado un bucket de Amazon Simple Storage Service y subido una imagen que desee utilizar como identificador a efectos de verificación de identidad.
- Ha seleccionado una segunda imagen para que sirva como imagen de destino para la verificación de identidad.

Creación de una colección

Para poder registrar un nuevo usuario o buscar un usuario en una colección o buscar un usuario, es necesario disponer de una colección con la que trabajar. Una colección de Amazon Rekognition es un contenedor del lado del servidor que se utiliza para almacenar información sobre los rostros detectados.

Crear la colección

El primer paso es escribir una función que cree una colección para que la utilice su aplicación. Amazon Rekognition almacena información sobre los rostros que se han detectado en contenedores del lado del servidor denominados Colecciones. Puede buscar rostros conocidos en la información facial almacenada en una colección. Para almacenar información facial, primero debe crear una colección mediante la operación `CreateCollection`.

1. Seleccione un nombre para la colección que le gustaría crear. En el código siguiente, sustituya el valor de `collection_id` por el nombre de la colección que desee crear y sustituya el valor de `region` por el nombre de la región definida en sus credenciales de usuario. Puede usar el argumento `Tags` para aplicar las etiquetas que desee a la colección, aunque no es obligatorio. La operación `CreateCollection` devolverá información sobre la colección que ha creado, incluido el ARN de la colección. Anote el ARN que reciba al ejecutar el código.

```
import boto3
```

```
def create_collection(collection_id, region):
    client = boto3.client('rekognition', region_name=region)

    # Create a collection
    print('Creating collection:' + collection_id)
    response = client.create_collection(CollectionId=collection_id,
    Tags={"SampleKey1":"SampleValue1"})
    print('Collection ARN: ' + response['CollectionArn'])
    print('Status code: ' + str(response['StatusCode']))
    print('Done...')

collection_id = 'collection-id-name'
region = "region-name"
create_collection(collection_id, region)
```

2. Guarde y ejecute el código. Copie el ARN de la colección.

   Ahora que se ha creado la colección de Rekognition, puede almacenar información facial e identificadores en esa colección. También podrá comparar los rostros con la información almacenada para su verificación.

Nuevo registro de usuarios

Querrá poder registrar nuevos usuarios y añadir su información a una colección. El proceso de registro de un nuevo usuario suele incluir los siguientes pasos:

Llamar a la operación **`DetectFaces`**

Escriba el código para comprobar la calidad de la imagen del rostro mediante la operación `DetectFaces`. Utilizará la operación `DetectFaces` para determinar si una imagen capturada por la cámara es adecuada para procesarla mediante la operación `SearchFacesByImage`. La imagen debe contener solo una cara. Proporcionará un archivo de imagen de entrada local a la operación `DetectFaces` y recibirá los detalles de los rostros detectados en la imagen. El siguiente código de ejemplo proporciona la imagen de entrada `DetectFaces` y, a continuación, comprueba si solo se ha detectado un rostro en la imagen.

1. En el siguiente ejemplo de código, sustituya `photo` por el nombre de la imagen de destino en la que quiere detectar rostros. También tendrá que sustituir el valor de `region` por el nombre de la región asociada a su cuenta.

```
import boto3
import json

def detect_faces(target_file, region):

    client=boto3.client('rekognition', region_name=region)

    imageTarget = open(target_file, 'rb')

    response = client.detect_faces(Image={'Bytes': imageTarget.read()},
    Attributes=['ALL'])

    print('Detected faces for ' + photo)
    for faceDetail in response['FaceDetails']:
        print('The detected face is between ' + str(faceDetail['AgeRange']['Low'])
              + ' and ' + str(faceDetail['AgeRange']['High']) + ' years old')

        print('Here are the other attributes:')
        print(json.dumps(faceDetail, indent=4, sort_keys=True))

        # Access predictions for individual face details and print them
        print("Gender: " + str(faceDetail['Gender']))
        print("Smile: " + str(faceDetail['Smile']))
        print("Eyeglasses: " + str(faceDetail['Eyeglasses']))
        print("Emotions: " + str(faceDetail['Emotions'][0]))

    return len(response['FaceDetails'])

photo = 'photo-name'
region = 'region-name'
face_count=detect_faces(photo, region)
print("Faces detected: " + str(face_count))

if face_count == 1:
    print("Image suitable for use in collection.")
else:
    print("Please submit an image with only one face.")
```

2. Guarde y ejecute el código de procedimiento.

Llamar a la operación **`CompareFaces`**

Su aplicación deberá poder registrar nuevos usuarios en una colección y confirmar la identidad de los usuarios recurrentes. Primero creará las funciones que se utilizan para registrar un nuevo usuario. Empezará por utilizar la `CompareFaces` operación para comparar una input/target imagen local del usuario y una ID/stored imagen. Si hay una coincidencia entre el rostro detectado en ambas imágenes, puede buscar en la colección para ver si el usuario está registrado en ella.

Comience por escribir una función que compare una imagen de entrada con la imagen de ID que ha almacenado en su bucket de Amazon S3. En el siguiente ejemplo de código, tendrá que proporcionar la imagen de entrada, que debería capturar después de utilizar algún tipo de detector de prueba de vida. También tendrá que pasar el nombre de una imagen almacenada en el bucket de Amazon S3.

1. Reemplace el valor de `bucket` por el nombre del bucket de Amazon S3 que contiene el archivo fuente. También tendrá que reemplazar el valor de `source_file` por el nombre de la imagen de origen que está utilizando. Reemplace el valor de `target_file` por el nombre del archivo de destino que ha proporcionado. Sustituya el valor de `region` de por el nombre de la `region` definida en sus credenciales de usuario.

   También querrá especificar un nivel mínimo de confianza en la coincidencia que se devuelve en la respuesta, utilizando el argumento `similarityThreshold`. Los rostros detectados solo se devolverán a la matriz `FaceMatches` si la confianza está por encima de este umbral. Su `similarityThreshold` elegido debe reflejar la naturaleza de su caso de uso específico. Cualquier caso de uso que involucre aplicaciones de seguridad críticas debe usar 99 como el umbral seleccionado.

```
import boto3

def compare_faces(bucket, sourceFile, targetFile, region):
    client = boto3.client('rekognition', region_name=region)

    imageTarget = open(targetFile, 'rb')

    response = client.compare_faces(SimilarityThreshold=99,
                                    SourceImage={'S3Object':
{'Bucket':bucket,'Name':sourceFile}},
                                    TargetImage={'Bytes': imageTarget.read()})

    for faceMatch in response['FaceMatches']:
        position = faceMatch['Face']['BoundingBox']
```

```
        similarity = str(faceMatch['Similarity'])
        print('The face at ' +
              str(position['Left']) + ' ' +
              str(position['Top']) +
              ' matches with ' + similarity + '% confidence')

    imageTarget.close()
    return len(response['FaceMatches'])

bucket = 'bucket-name'
source_file = 'source-file-name'
target_file = 'target-file-name'
region = "region-name"
face_matches = compare_faces(bucket, source_file, target_file, region)
print("Face matches: " + str(face_matches))

if str(face_matches) == "1":
    print("Face match found.")
else:
    print("No face match found.")
```

2. Guarde y ejecute el código de procedimiento.

   Se le devolverá un objeto de respuesta que contiene información sobre el rostro coincidente y el nivel de confianza.

Llamar a la operación **SearchFacesByImage**

Si el nivel de confianza de la operación `CompareFaces` es superior al `SimilarityThreshold` que ha elegido, querrá buscar en su colección un rostro que pueda coincidir con la imagen introducida. Si se encuentra una coincidencia en su colección, significa que es probable que el usuario ya esté registrado en la colección y que no sea necesario registrar un nuevo usuario en la colección. Si no hay ninguna coincidencia, puede registrar al nuevo usuario en su colección.

1. Comience por escribir el código que invocará la operación `SearchFacesByImage`. La operación tomará como argumento un archivo de imagen local y, a continuación, buscará en su `Collection` un rostro que coincida con los rostros más grandes detectados en la imagen proporcionada.

   En el siguiente ejemplo de código, cambie el valor de `collectionId` a la colección en la que desee buscar. Sustituya el valor de `region` por el nombre de la región asociada a su

cuenta. También tendrá que reemplazar el valor de `photo` por el nombre del archivo de entrada. También querrá especificar un umbral de similitud sustituyendo el valor de `threshold` por un percentil elegido.

```
import boto3

collectionId = 'collection-id-name'
region = "region-name"
photo = 'photo-name'
threshold = 99
maxFaces = 1
client = boto3.client('rekognition', region_name=region)

# input image should be local file here, not s3 file
with open(photo, 'rb') as image:
    response = client.search_faces_by_image(CollectionId=collectionId,
    Image={'Bytes': image.read()},
    FaceMatchThreshold=threshold, MaxFaces=maxFaces)

faceMatches = response['FaceMatches']
print(faceMatches)

for match in faceMatches:
    print('FaceId:' + match['Face']['FaceId'])
    print('ImageId:' + match['Face']['ImageId'])
    print('Similarity: ' + "{:.2f}".format(match['Similarity']) + "%")
    print('Confidence: ' + str(match['Face']['Confidence']))
```

2. Guarde y ejecute el código de procedimiento. Si ha habido una coincidencia, significa que la persona reconocida en la imagen ya forma parte de la colección y no es necesario continuar con los siguientes pasos. En este caso, solo tiene que permitir que el usuario acceda a la aplicación.

Llamar a la operación **`IndexFaces`**

Suponiendo que no se haya encontrado ninguna coincidencia en la colección que ha buscado, querrá añadir el rostro del usuario a su colección. Para ello, llame a la operación `IndexFaces`. Cuando llamas IndexFaces, Amazon Rekognition extrae los rasgos faciales de un rostro identificado en la imagen de entrada y guarda los datos en la colección especificada.

1. Comience por escribir el código para llamar a `IndexFaces`. Sustituya el valor `image` de por el nombre del archivo local que desee utilizar como imagen de entrada para la operación.

IndexFaces También tendrá que sustituir el valor de `photo_name` por el nombre deseado para la imagen de entrada. Asegúrese de reemplazar el valor de `collection_id` por el ID de la colección que ha creado anteriormente. Después, sustituya el valor de `region` por el nombre de la región asociada a su cuenta. También querrá especificar un valor para el parámetro de entrada `MaxFaces`, que define el número máximo de caras de una imagen que se deben indexar. El valor predeterminado para este parámetro es 1.

```
import boto3

def add_faces_to_collection(target_file, photo, collection_id, region):
    client = boto3.client('rekognition', region_name=region)

    imageTarget = open(target_file, 'rb')

    response = client.index_faces(CollectionId=collection_id,
                                  Image={'Bytes': imageTarget.read()},
                                  ExternalImageId=photo,
                                  MaxFaces=1,
                                  QualityFilter="AUTO",
                                  DetectionAttributes=['ALL'])
    print(response)

    print('Results for ' + photo)
    print('Faces indexed:')
    for faceRecord in response['FaceRecords']:
        print('  Face ID: ' + faceRecord['Face']['FaceId'])
        print('  Location: {}'.format(faceRecord['Face']['BoundingBox']))
        print('  Image ID: {}'.format(faceRecord['Face']['ImageId']))
        print('  External Image ID: {}'.format(faceRecord['Face']
['ExternalImageId']))
        print('  Confidence: {}'.format(faceRecord['Face']['Confidence']))

    print('Faces not indexed:')
    for unindexedFace in response['UnindexedFaces']:
        print(' Location: {}'.format(unindexedFace['FaceDetail']['BoundingBox']))
        print(' Reasons:')
        for reason in unindexedFace['Reasons']:
            print('   ' + reason)
    return len(response['FaceRecords'])

image = 'image-file-name'
collection_id = 'collection-id-name'
```

```
photo_name = 'desired-image-name'
region = "region-name"

indexed_faces_count = add_faces_to_collection(image, photo_name, collection_id,
 region)
print("Faces indexed count: " + str(indexed_faces_count))
```

2. Guarde y ejecute el código de procedimiento. Determine si desea guardar alguno de los datos devueltos por la operación `IndexFaces`, como el FaceID asignado a la persona de la imagen. En la siguiente sección se examinará cómo guardar estos datos. Copie los valores `FaceId`, `ImageId` y `Confidence` devueltos antes de continuar.

Guardar y almacenar datos de imagen y de FaceID en Amazon S3 y Amazon DynamoDB

Una vez obtenido el ID de rostro de la imagen de entrada, los datos de la imagen se pueden guardar en Amazon S3, mientras que los datos faciales y la URL de la imagen se pueden introducir en una base de datos como DynamoDB.

1. Guarde y cargue el código para subir la imagen de entrada en la base de datos de Amazon S3. En el ejemplo de código que aparece a continuación, sustituya el valor de `bucket` por el nombre del bucket en el que desea subir el archivo y, a continuación, sustituya el valor de `file_name` por el nombre del archivo local que desea almacenar en su bucket de Amazon S3. Proporcione un nombre clave que identifique el archivo en el bucket de Amazon S3 sustituyendo el valor de `key_name` por el nombre que desee asignar al archivo de imagen. El archivo que desea cargar es el mismo que se definió en los ejemplos de código anteriores, que es el archivo de entrada que utilizó IndexFaces. Por último, sustituya el valor de `region` por el nombre de la región asociada a su cuenta.

```
import boto3
import logging
from botocore.exceptions import ClientError

# store local file in S3 bucket
bucket = "amzn-s3-demo-bucket"
file_name = "file-name"
key_name = "key-name"
region = "region-name"
s3 = boto3.client('s3', region_name=region)
# Upload the file
```

```
try:
    response = s3.upload_file(file_name, bucket, key_name)
    print("File upload successful!")
except ClientError as e:
    logging.error(e)
```

2. Guarde y ejecute el ejemplo de código correspondiente para subir la imagen de entrada en Amazon S3.

3. También querrá guardar el ID de rostro devuelto en una base de datos. Para ello, puede crear una tabla de base de datos de DynamoDB y, a continuación, subir el ID de rostro en esa tabla. En el siguiente ejemplo de código se crea una tabla de DynamoDB. Tenga en cuenta que solo necesita ejecutar el código que crea esta tabla una vez. En el código siguiente, sustituya el valor de `region` por el valor de la región asociada a su cuenta. También tendrá que sustituir el valor de `database_name` por el nombre que desee asignar a la tabla de DynamoDB.

```
import boto3

# Create DynamoDB database with image URL and face data, face ID

def create_dynamodb_table(table_name, region):
    dynamodb = boto3.client("dynamodb", region_name=region)

    table = dynamodb.create_table(
        TableName=table_name,
        KeySchema=[{
                'AttributeName': 'FaceID', 'KeyType': 'HASH'  # Partition key
                },],
            AttributeDefinitions=[
            {
                'AttributeName': 'FaceID', 'AttributeType': 'S'  }, ],
                ProvisionedThroughput={
            'ReadCapacityUnits': 10, 'WriteCapacityUnits': 10  }
    )
    print(table)
    return table

region = "region-name"
database_name = 'database-name'
dynamodb_table = create_dynamodb_table(database_name, region)
print("Table status:", dynamodb_table)
```

4. Guarde y ejecute el código siguiente para crear la tabla.

5. Tras crear la tabla, puede cargar en ella FaceId lo devuelto. Para ello, deberá establecer una conexión con la tabla con la función Table y, a continuación, utilizar la función `put_item` para subir los datos.

   En el siguiente ejemplo de código, sustituya el valor de `bucket` por el nombre del bucket que contiene la imagen de entrada que ha subido a Amazon S3. También tendrá que sustituir el valor de `file_name` por el nombre del archivo de entrada que has subido a su bucket de Amazon S3 y el valor de `key_name` por la clave que utilizó anteriormente para identificar el archivo de entrada. Por último, sustituya el valor de `region` por el nombre de la región asociada a su cuenta. Estos valores deben coincidir con los proporcionados en el paso 1.

   `AddDBEntry`Almacena los valores de confianza FaceId ImageId, y asignados a un rostro de una colección. Proporcione a la siguiente función los valores que guardó durante el paso 2 de la sección `IndexFaces` anterior.

```
import boto3
from pprint import pprint
from decimal import Decimal
import json

# The local file that was stored in S3 bucket
bucket = "amzn-s3-demo-bucket"
file_name = "file-name"
key_name = "key-name"
region = "region-name"
# Get URL of file
file_url = "https://s3.amazonaws.com/{}/{}".format(bucket, key_name)
print(file_url)

# upload face-id, face info, and image url
def AddDBEntry(file_name, file_url, face_id, image_id, confidence):
    dynamodb = boto3.resource('dynamodb', region_name=region)
    table = dynamodb.Table('FacesDB-4')
    response = table.put_item(
       Item={
            'ExternalImageID': file_name,
            'ImageURL': file_url,
            'FaceID': face_id,
            'ImageID': image_id,
            'Confidence': json.loads(json.dumps(confidence), parse_float=Decimal)
       }
    )
```

```
    return response

# Mock values for face ID, image ID, and confidence - replace them with actual
 values from your collection results
dynamodb_resp = AddDBEntry(file_name, file_url, "FACE-ID-HERE",
    "IMAGE-ID-HERE", confidence-here)
print("Database entry successful.")
pprint(dynamodb_resp, sort_dicts=False)
```

6. Guarde y ejecute el ejemplo de código siguiente para almacenar los datos de ID de rostro devueltos en una tabla.

Inicio de sesión de usuario existente

Una vez que un usuario se ha registrado en una colección, se puede autenticar cuando vuelva mediante la operación `SearchFacesByImage`. Necesitará obtener una imagen de entrada y, a continuación, comprobar la calidad de la imagen de entrada utilizando `DetectFaces`. Esto determina si se ha utilizado una imagen adecuada antes de ejecutar la operación `SearchFacesbyImage`.

Llame a la DetectFaces operación

1. Utilizará la operación `DetectFaces` para comprobar la calidad de la imagen del rostro y determinar si una imagen capturada por la cámara es adecuada para ser procesada por la operación `SearchFacesByImage`. La imagen de entrada debe contener solo un rostro. En el siguiente ejemplo de código se toma una imagen de entrada que se proporciona a la operación `DetectFaces`.

   En el siguiente ejemplo de código, sustituya el valor de `photo` por el nombre de la imagen de destino local y sustituya el valor de `region` por el nombre de la región asociada a su cuenta.

```
import boto3
import json

def detect_faces(target_file, region):

    client=boto3.client('rekognition', region_name=region)

    imageTarget = open(target_file, 'rb')
```

```
    response = client.detect_faces(Image={'Bytes': imageTarget.read()},
    Attributes=['ALL'])

    print('Detected faces for ' + photo)
    for faceDetail in response['FaceDetails']:
        print('The detected face is between ' + str(faceDetail['AgeRange']['Low'])
              + ' and ' + str(faceDetail['AgeRange']['High']) + ' years old')

        print('Here are the other attributes:')
        print(json.dumps(faceDetail, indent=4, sort_keys=True))

        # Access predictions for individual face details and print them
        print("Gender: " + str(faceDetail['Gender']))
        print("Smile: " + str(faceDetail['Smile']))
        print("Eyeglasses: " + str(faceDetail['Eyeglasses']))
        print("Emotions: " + str(faceDetail['Emotions'][0]))

    return len(response['FaceDetails'])

photo = 'photo-name'
region = 'region-name'
face_count=detect_faces(photo, region)
print("Faces detected: " + str(face_count))

if face_count == 1:
    print("Image suitable for use in collection.")
else:
    print("Please submit an image with only one face.")
```

2. Guarde y ejecute el código.

Llama a la SearchFacesByImage operación

1. Escriba el código para comparar el rostro detectado con los rostros de la colección con `SearchFacesByImage`. Utilizará el código que se muestra en la siguiente sección de registro de nuevos usuarios y proporcionará la imagen de entrada para la operación `SearchFacesByImage`.

   En el siguiente ejemplo de código, cambie el valor de `collectionId` por la colección en la que desee buscar. También cambiará el valor de `bucket` por el nombre de un bucket de Amazon S3 y el valor de `fileName` por un archivo de imagen de ese bucket. Sustituya el valor de `region`

por el nombre de la región asociada a su cuenta. También querrá especificar un umbral de similitud sustituyendo el valor de `threshold` por un percentil elegido.

```
import boto3

bucket = 'amzn-s3-demo-bucket'
collectionId = 'collection-id-name'
region = "region-name"
fileName = 'file-name'
threshold = 70
maxFaces = 1
client = boto3.client('rekognition', region_name=region)

# input image should be local file here, not s3 file
with open(fileName, 'rb') as image:
    response = client.search_faces_by_image(CollectionId=collectionId,
    Image={'Bytes': image.read()},
    FaceMatchThreshold=threshold, MaxFaces=maxFaces)
```

2. Guarde y ejecute el código.

Comprobar el FaceId devuelto y el nivel de confianza

Ahora puede comprobar la información sobre las coincidencias FaceId imprimiendo elementos de respuesta como los FaceId atributos de similitud y confianza.

```
faceMatches = response['FaceMatches']
print(faceMatches)

for match in faceMatches:
    print('FaceId:' + match['Face']['FaceId'])
    print('ImageId:' + match['Face']['ImageId'])
    print('Similarity: ' + "{:.2f}".format(match['Similarity']) + "%")
    print('Confidence: ' + str(match['Face']['Confidence']))
```

Uso de Amazon Rekognition (REK) para detectar etiquetas de aplicaciones de marketing

Este tutorial le guía a través del proceso de creación de una aplicación Python de muestra que puede usarse para enviar correos electrónicos a las personas en función de las imágenes cargadas en un

sitio web. La aplicación de muestra está diseñada para promover la participación de los usuarios en una campaña de marketing en las redes sociales mediante el envío a estos de correos electrónicos personalizados sobre ofertas de viajes si se reconocen referencias en sus fotos.

La solución integra varios AWS servicios, incluidos Amazon Rekognition, Amazon S3 CloudWatch, DynamoDB y Amazon SES. La aplicación utiliza Amazon Rekognition para detectar etiquetas en las imágenes cargadas por cuentas reconocidas en una base de datos de DynamoDB; después, se envía al titular de la cuenta un correo electrónico de marketing basado en las etiquetas detectadas en la imagen. La arquitectura completa de la solución es la siguiente:

- Almacene los datos de los usuarios en la base de datos de DynamoDB.
- El usuario carga metadatos (número de cuenta de usuario) y datos de las imágenes en Amazon S3.
- Una función Lambda se invoca DetectLabelspara identificar y registrar las etiquetas de la imagen cargada, buscar el correo electrónico del usuario en DynamoDB y enviar un correo electrónico de marketing al usuario que ha subido la imagen.
- Los resultados de la función Lambda se registran CloudWatch para su posterior revisión.

A continuación, se proporciona información general sobre todos los pasos del tutorial:

1. Crear y rellenar una tabla de DynamoDB
2. Escribir una función de Lambda con registro y notificación
3. Configurar la función de Lambda y los permisos
4. Cargar imágenes y metadatos con Amazon S3
5. CloudWatch Registros de encuestas.

Temas

- Requisitos previos
- Cómo crear y rellenar una tabla de DynamoDB
- Creación de una función de Lambda con registro y notificación
- Configuración de los permisos y la función de Lambda
- Carga de imágenes en Amazon S3 con metadatos
- CloudWatch Registros de sondeos del cliente

Requisitos previos

Antes de comenzar este tutorial, necesita lo siguiente:

- Una AWS cuenta y los permisos de IAM adecuados.
- Python y Boto3 instalados en el entorno de desarrollo
- Conocimientos básicos de Lambda, Amazon S3, DynamoDB, Amazon SES y. CloudWatch
- Conocimientos básicos del uso de Lambda y Amazon SES

Cómo crear y rellenar una tabla de DynamoDB

Para comenzar este tutorial, crearemos una tabla de DynamoDB para almacenar los datos de los clientes, como la dirección de correo electrónico, la edad, el número de teléfono y el estado de la membresía, utilizando una variable AccountNumber «» como clave principal. También insertaremos algunos datos de muestra en la tabla.

Creación de la tabla de DynamoDB

En primer lugar, vamos a crear la tabla de DynamoDB. Aquí se indica cómo configurarla con Python y Boto3. Creamos una función llamada `create_user_table` y nos conectamos al recurso de DynamoDB que contiene. En la muestra de código que aparece a continuación, sustituya el valor de “region_name” por el nombre de la región en la que opera su cuenta y, a continuación, ejecute la celda de código para crear la tabla.

```
import boto3
def create_user_table(dynamodb=None):
    if not dynamodb:
        dynamodb = boto3.resource('dynamodb', region_name='us-east-1')

    table = dynamodb.create_table(
        TableName='CustomerDataTable',
        KeySchema=[
            {
                'AttributeName': 'AccountNumber',
                'KeyType': 'HASH'  # Partition key
            },
        ],
        AttributeDefinitions=[
```

```
            {
                'AttributeName': 'AccountNumber',
                'AttributeType': 'S'
            },
        ],
        ProvisionedThroughput={
            'ReadCapacityUnits': 10,
            'WriteCapacityUnits': 10
        }
    )

    # Wait until the table exists.
    table.wait_until_exists()
    print("Table status:", table.table_status)

# Create the DynamoDB table.
create_user_table()
```

Al ejecutar este script, se ha configurado una tabla de DynamoDB llamada `CustomerDataTable` con `AccountNumber` como clave principal.

Inserción de datos de muestra

Ahora vamos a insertar algunos datos de muestra en la tabla. Estos datos de muestra nos ayudarán a probar la funcionalidad completa de la aplicación.

Crearemos una segunda función que agregue datos de muestra a la misma `CustomerDataTable` que hemos creado anteriormente. Con el siguiente código se crean tres entradas de muestra y cada una incluye un número de cuenta, una dirección de correo electrónico, una edad, un número de teléfono y el estado de membresía. En la muestra de código que aparece a continuación, sustituya el valor de `region_name` por el nombre de la región en la que opera su cuenta y, a continuación, ejecute la celda de código para crear la tabla. Si quiere probar la parte de entrega de correo electrónico de la aplicación, sustituya el valor de `EmailAddress` en la primera entrada del cliente a continuación por una dirección de correo electrónico en la que pueda recibir correo de este tipo. Guarde y ejecute el código.

```
import boto3

def insert_sample_data(dynamodb=None):
```

```
    if not dynamodb:
        dynamodb = boto3.resource('dynamodb', region_name='us-east-1')

    table = dynamodb.Table('CustomerDataTable')

    # Sample data
    customers = [
        {
            'AccountNumber': 'ACC1000',
            'EmailAddress': 'email-for-delivery-here',
            'Age': 30,
            'PhoneNumber': '123-456-7890',
            'MembershipStatus': 'Active'
        },
        {
            'AccountNumber': 'ACC1001',
            'EmailAddress': 'jane.doe@example.com',
            'Age': 25,
            'PhoneNumber': '098-765-4321',
            'MembershipStatus': 'Inactive'
        },
        {
            'AccountNumber': 'ACC1002',
            'EmailAddress': 'pat.candella@example.com',
            'Age': 35,
            'PhoneNumber': '555-555-5555',
            'MembershipStatus': 'Active'
        }
    ]
        # Inserting data
    for customer in customers:
            print(f"Adding customer: {customer['AccountNumber']}")
            table.put_item(Item=customer)

# Insert sample data into DynamoDB
insert_sample_data()
```

Una vez que la tabla de DynamoDB se haya configurado y rellenado, podemos proceder a integrar esta recuperación de datos en la función de Lambda. Esto permite a nuestra aplicación recuperar los detalles del usuario en función del número de cuenta que se identificará en el próximo flujo de trabajo de procesamiento de imágenes.

Creación de una función de Lambda con registro y notificación

Ahora, podemos crear una función de Lambda. Queremos asegurarnos de que la función de Lambda, que se desencadena al cargar la imagen, pueda leer tanto los datos de imagen como los metadatos de la cuenta y usarlos para realizar una búsqueda en la tabla DynamoDB de los datos de usuario asociados. Esto significa que no solo tendremos que invocar la función de Amazon Rekognition, sino que también necesitaremos una DetectLabels función que la tome AccountNumber y la utilice para recuperar la dirección de correo electrónico asociada de DynamoDB. Asimismo, necesitamos una función que envíe un correo electrónico a la dirección correspondiente que utiliza Amazon SES. Por último, registramos la información sobre el proceso mediante un registrador. Estos datos se mostrarán mediante. CloudWatch

Creación de la función de Lambda

A continuación, se describe cómo puede gestionar una función de Lambda estos requisitos. En el siguiente ejemplo de código, asegúrese de que se ha especificado la tabla de DynamoDB correcta si ha utilizado algo CustomerDataTable distinto del nombre de la tabla. Además, en la función “send_marketing_email”, debe reemplazar el valor del argumento “Source” por una dirección de correo electrónico a la que tenga acceso y que funcione como correo electrónico de envío.

```
import json
import boto3
import logging

logger = logging.getLogger()
logger.setLevel(logging.INFO)

def lambda_handler(event, context):
    s3_bucket = event['Records'][0]['s3']['bucket']['name']
    s3_object_key = event['Records'][0]['s3']['object']['key']
    print(s3_bucket)
    print(s3_object_key)

    s3 = boto3.client('s3')
    try:
        s3_response = s3.head_object(Bucket=s3_bucket, Key=s3_object_key)
        account_number = s3_response['Metadata']['account_number']
    except Exception as e:
        logger.error(f"Failed to retrieve object or metadata: {str(e)}")
        raise e  # Optionally re-raise to handle the error upstream or signal a failure
```

```
    rekognition = boto3.client('rekognition')
    try:
        labels_response = rekognition.detect_labels(Image={'S3Object': {'Bucket':
 s3_bucket, 'Name': s3_object_key}})
        #logger.info(f"Detected labels: {json.dumps(labels_response['Labels'],
 indent=4)}")
    except Exception as e:
        #logger.info(f"Detected label: {label['Name']}")
        raise e

    #logger.info(f"Detected labels: {json.dumps(labels_response['Labels'], indent=4)}")

    landmark_detected = any(label['Name'] == 'Landmark' and label['Confidence'] > 20
 for label in labels_response['Labels'])
    if landmark_detected:
        result = notify_user_based_on_landmark(account_number)
        print(result)
        #logger.info(f"Detected label: {label['Name']}")
        #logger.info(f"Notification sent: {result}")
    return {
        'statusCode': 200,
        'body': json.dumps('Process completed successfully!')
    }

def notify_user_based_on_landmark(account_number):
     # Retrieve user data from DynamoDB
    dynamodb = boto3.resource('dynamodb')
    table = dynamodb.Table('CustomerDataTable')
    user_info = table.get_item(Key={'AccountNumber': account_number})

    # Send email if user is found
    if 'Item' in user_info:
        send_marketing_email(user_info['Item']['EmailAddress'])
    return "User notified with relevant travel information."

def send_marketing_email(email):
    ses = boto3.client('ses')
    response = ses.send_email(
        Source='your-email@example.com',
        Destination={'ToAddresses': [email]},
        Message={
            'Subject': {'Data': 'Explore New Destinations!'},
            'Body': {
```

```
                'Text': {'Data': 'Check out our exclusive travel packages inspired by
 the landmark in your uploaded image!'}
            }
        }
    )
    return f"Email sent to {email} with status {response['ResponseMetadata']
['HTTPStatusCode']}"
    print("succeess")
```

Ahora que hemos escrito la función Lambda, necesitamos configurar los permisos para Lambda y crear una instancia de nuestra función Lambda en la consola de administración. AWS

Configuración de los permisos y la función de Lambda

Creación o actualización del rol de IAM

Para poder utilizar una función de Lambda a fin de gestionar cualquier imagen que hayan cargado los usuarios, antes debemos gestionar los permisos correspondientes a esta. La función Lambda necesita una función de IAM con un ámbito específico que tenga políticas que le permitan interactuar con Amazon S3, DynamoDB, Amazon SES, Amazon Rekognition y. CloudWatch

Para configurar el rol de IAM para la función de Lambda:

1. AWS Consola de administración abierta.
2. Vaya a IAM > Roles > Crear rol.

   Elija “Crear rol”.
3. Seleccione Lambda como el servicio que va a usar este rol. Haga clic en “Siguiente”.
4. Adjunte las siguientes políticas (tenga en cuenta que estas políticas se seleccionan únicamente con fines de demostración; en un entorno de producción real, le interesa limitar los permisos solo a los que sean necesarios):

   - Amazon S3 ReadOnlyAccess
   - AmazonRekognitionReadOnlyAccess
   - AmazonDynamoDBFullAcceso
   - SESFullAcceso a Amazon
   - AWSLambdaEjecutar

- AWSLambdaBasicExecutionRole (para CloudWatch registrar)

Haga clic en “Siguiente”.

5. Asigne un nombre y una descripción al rol y, a continuación, seleccione “Crear rol” para crearlo.
6. Una vez que hayamos configurado los permisos adecuados para Lambda, podemos crear una instancia de la función Lambda mediante la consola de administración. AWS

   En la Consola de administración de AWS, vaya al servicio Lambda.
7. Haga clic en “Crear una función”. Elija “Crear desde cero”.
8. Escriba un nombre para la función y seleccione el entorno en tiempo de ejecución Python. En “Cambiar el rol de ejecución predeterminado”, seleccione “Uso de un rol existente” y, a continuación, elija el rol de IAM que ha creado anteriormente.
9. Ahora tiene que crear y actualizar el código en la pestaña de la función de Lambda. Vaya a la pestaña llamada “lambda_function” y reemplace el código que aparece en esta por la muestra de código de Lambda que proceda.

   Guarde los cambios e impleméntelos.
10. Ahora tiene que configurar un evento de Amazon S3 como desencadenador de Lambda.

    En la página o la pestaña de configuración de la función de Lambda, vaya a “Desencadenador” y haga clic en “Agregar desencadenador”.
11. Seleccione Amazon S3 en la lista de desencadenadores disponibles.
12. Para configurar el desencadenador, seleccione el bucket desde el que debe desencadenarse la función.

    Elija el tipo de evento, PUT. De forma opcional, puede especificar un prefijo o un sufijo si solo quiere procesar archivos con determinados nombres o tipos.
13. Haga clic en “Agregar” para activar el desencadenador y guarde la configuración.

Verificación de direcciones de correo electrónico de Amazon SES

Para poder usar Amazon SES a fin de enviar mensajes de correo electrónico, antes debe verificar las direcciones de correo electrónico tanto del remitente como del destinatario. Para ello:

1. Vaya a la consola de Amazon SES. Vaya a “Administración de identidades” y, a continuación, a “Direcciones de correo electrónico”.

2. Haga clic en “Verificar una dirección de email nueva”. Escriba la dirección de correo electrónico que quiere verificar y, a continuación, elija “Verificar esta dirección de correo electrónico”. Recibirá un correo electrónico con un enlace de verificación. Haga clic en el enlace para completar el proceso de verificación.
3. Una vez que haya verificado las direcciones de correo electrónico de ambas cuentas, haya configurado la función de Lambda con los permisos correctos y haya creado algunos datos de clientes de muestra, puede probar la función de Lambda si carga una imagen de prueba en el bucket de Amazon S3 que elija.

Carga de imágenes en Amazon S3 con metadatos

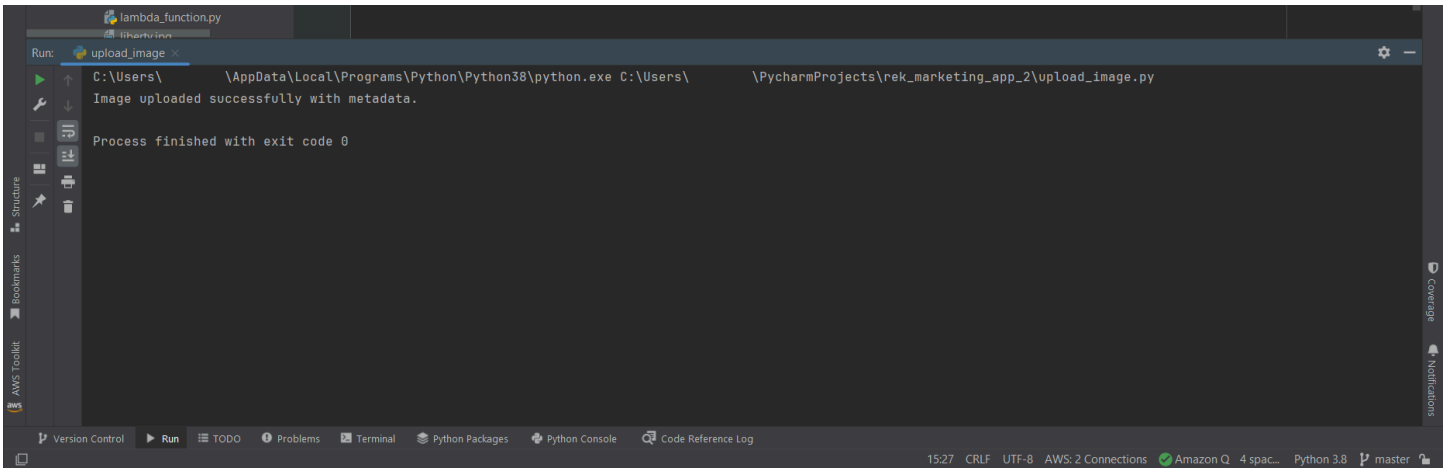
Ahora podemos cargar una imagen de prueba en el bucket de Amazon S3 que hayamos elegido mediante la AWS consola o el script que preparamos anteriormente. Escriba un script que cargue una imagen en el bucket de Amazon S3 que ha especificado anteriormente en la función de Lambda. En el ejemplo de código que aparece a continuación, especifique la ruta de acceso de la imagen como primer argumento de “upload_image_to_s3”, el nombre del bucket como segundo argumento y el número de cuenta del usuario que carga la imagen como argumento final.

```
import boto3
def upload_image_to_s3(file_name, bucket, account_number):
    s3 = boto3.client('s3')
    try:
        with open(file_name, 'rb') as data:
            s3.upload_fileobj(
                Fileobj=data,
                Bucket=bucket,
                Key=file_name,
                ExtraArgs={
                    'Metadata': {'account_number': account_number}
                }
            )
        print("Image uploaded successfully with metadata.")
    except Exception as e:
        print("Failed to upload image")
        print(e)

# Usage
upload_image_to_s3('path-to-image-here', 's3-bucket-name-here', 'user-account-number-
here')
```

En esta función, se utiliza el parámetro `ExtraArgs` del método `upload_fileobj` para incluir metadatos definidos por el usuario (`account_number`) junto con la imagen. Estos metadatos se pueden utilizar posteriormente AWS para procesar la imagen en consecuencia.

Guarde y ejecute el script. Al hacerlo, se cargará la imagen.



Unos minutos después de cargar la imagen, debería recibir un correo electrónico en la dirección que ha asociado anteriormente a la cuenta especificada aquí.

CloudWatch Registros de sondeos del cliente

Compruebe los CloudWatch registros de la función Lambda para ver si se activó y ejecutó según lo previsto. Los registros se encuentran en > Registros > Grupos de registros CloudWatch >AWS/lambda/your_function_name. También puede escribir un script para acceder a los registros e imprimirlos. La siguiente muestra de código sondea los registros del grupo Lambda e imprime aquellos que se han generado en la última hora. Guarde y ejecute el código.

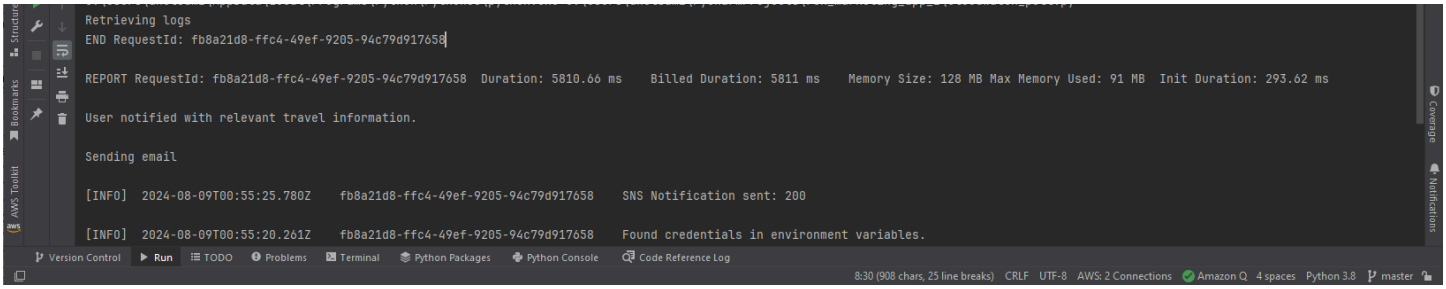
```
import boto3
import time

def fetch_lambda_logs(log_group_name, start_time):
    client = boto3.client('logs')
    query = "fields @timestamp, @message | sort @timestamp desc | limit 20"
    start_query_response = client.start_query(
        logGroupName=log_group_name,
        startTime=int(start_time),
        endTime=int(time.time()),
```

```
        queryString=query,
    )
    query_id = start_query_response['queryId']
    response = None
    while response is None or response['status'] == 'Running':
        time.sleep(1)  # Wait for 1 second before checking the query status again
        response = client.get_query_results(queryId=query_id)
    return response['results']

# Usage
log_group = '/aws/lambda/RekMediaFunction'
logs = fetch_lambda_logs(log_group, int(time.time()) - 3600)  # Fetch logs from the
 last hour
print("Retrieving logs")
for log in logs:
    #print(log)
    print(log[1]['value'])
```

Al ejecutar el código, deberían imprimirse los registros y debería poder ver que el usuario ha recibido una notificación por correo electrónico con la información de viaje pertinente.



Ha creado correctamente una aplicación capaz de detectar etiquetas en las imágenes cargadas en un bucket de Amazon S3 y de enviar, después, por correo electrónico un mensaje promocional al usuario que ha cargado la imagen. Asegúrese de eliminar los recursos que ya no necesite para que no se le cobre por ellos.

Detección de etiquetas en una imagen mediante Lambda y Python

AWS Lambda es un servicio informático que puede utilizar para ejecutar código sin aprovisionar ni administrar servidores. Puede llamar a las operaciones de la API de Rekognition desde una función de Lambda. En las instrucciones siguientes se explica cómo crear una función de Lambda en Python que llame a `DetectLabels`.

La función de Lambda llama a `DetectLabels` y devuelve una matriz de etiquetas detectadas en la imagen y el nivel de confianza por el que se detectan.

Las instrucciones incluyen un ejemplo de código de Python que muestra cómo llamar a la función de Lambda y proporcionarle una imagen procedente de un bucket de Amazon S3 o de su ordenador local.

Asegúrese de que las imágenes que ha elegido cumplen los límites de Rekognition. Consulte las directrices y las cuotas en Rekognition y en la referencia de la API para obtener información sobre DetectLabels los límites de tamaño y tipo de archivo de imagen.

Creación de una función de Lambda (consola)

En este paso, se crea una función de Lambda vacía y una función de ejecución de IAM que permita a la función llamar a la operación `DetectLabels`. En pasos posteriores, añadirá el código fuente y, si lo desea, una capa a la función de Lambda.

Si utiliza documentos almacenados en un bucket de Amazon S3, en este paso también se muestra cómo conceder acceso al bucket que almacena los documentos.

Para crear una AWS Lambda función (consola)

1. Inicie sesión en Consola de administración de AWS y abra la AWS Lambda consola en https://console.aws.amazon.com/lambda/.
2. Seleccione Creación de función. Para obtener más información, consulte Crear una función de Lambda con la consola.
3. Elija las siguientes opciones:
   - Elija Crear desde cero.
   - Ingrese un valor en Nombre de la función.
   - Para Tiempo de ejecución, elige la versión más reciente de Python.
   - En Arquitectura, elija x86_64.
4. Elija Crear función para crear la función de AWS Lambda .
5. En la página de la función, seleccione la pestaña Configuración.
6. En el panel Permisos, en Rol de ejecución, elija el nombre del rol para abrir el rol en la consola de IAM.
7. En la pestaña Permisos, elija Añadir permisos y después Crear política insertada.

8. Elija la pestaña JSON y reemplace la política por la siguiente política:

   JSON

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

9. Elija Revisar política.
10. Introduzca un nombre para la política, por ejemplo DetectLabels-access.
11. Elija Crear política.
12. Si va a almacenar documentos para analizarlos en un bucket de Amazon S3, debe añadir una política de acceso a Amazon S3. Para ello, repita los pasos 7 a 11 en la consola de AWS Lambda y realice los siguientes cambios.

    a. En el paso 8, utilice la siguiente política. *bucket/folder path*Sustitúyala por la ruta del depósito y la carpeta de Amazon S3 a los documentos que desee analizar.

       JSON

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

b. En el paso 10, elija un nombre de política diferente, como S3Bucket-access.

(Opcional) Crear una capa (consola)

No es necesario realizar este paso para utilizar una función de Lambda y realizar una llamada a `DetectLabels`.

La `DetectLabels` operación se incluye en el entorno Lambda Python predeterminado como parte del AWS SDK para Python (Boto3).

Si otras partes de la función Lambda requieren actualizaciones de AWS servicio recientes que no están en el entorno Lambda Python predeterminado, puede realizar este paso para añadir la versión más reciente del SDK de Boto3 como capa a su función.

Para añadir SDK como una capa, primero se crea un archivo de archivo .zip con el SDK de Boto3. A continuación, tendrá que crear una capa y añadir el archivo de zip a la capa. Para obtener más información, consulte Uso de capas de Lambda.

Cómo crear y añadir una capa (consola)

1. Abra una línea de comandos e introduzca los siguientes comandos para crear un paquete de despliegue con la versión más reciente del SDK. AWS

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

2. Anote el nombre del archivo zip (boto3-layer.zip), que utilizará en el paso 8 de este procedimiento.
3. Abre la AWS Lambda consola en https://console.aws.amazon.com/lambda/.
4. En el panel de navegación, elija Capas.
5. Elija Crear capa.
6. Introduzca valores para Nombre y Descripción.
7. Para Tipo de entrada de código, elija Cargar un archivo .zip y seleccione Cargar.
8. En el cuadro de diálogo, elija el archivo .zip (boto3-layer.zip) que creó en el paso 1 de este procedimiento.
9. Para Tiempo de ejecución compatibles, elige la versión más reciente de Python.

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 anteriormente en ???.
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.

Añadir código Python (consola)

En este paso, añada código Python a la función de Lambda mediante el editor de código de la consola de Lambda. El código detecta las etiquetas de una imagen mediante la operación `DetectLabels`. Devuelve una matriz de etiquetas detectadas en la imagen, así como el nivel de confianza en las etiquetas detectadas.

El documento que proporcione a la operación `DetectLabels` puede estar ubicado en un bucket de Amazon S3 o en un ordenador local.

Cómo añadir código Python (consola)

1. Navegue hasta la pestaña Código.
2. En el editor de código, sustituya el código del archivo lambda_function.py por el código siguiente:

```
import boto3
import logging
from botocore.exceptions import ClientError
import json
import base64

# Instantiate logger
logger = logging.getLogger(__name__)

# connect to the Rekognition client
rekognition = boto3.client('rekognition')
```

```
def lambda_handler(event, context):

    try:
        image = None
        if 'S3Bucket' in event and 'S3Object' in event:
            s3 = boto3.resource('s3')
            s3_object = s3.Object(event['S3Bucket'], event['S3Object'])
            image = s3_object.get()['Body'].read()

        elif 'image' in event:
            image_bytes = event['image'].encode('utf-8')
            img_b64decoded = base64.b64decode(image_bytes)
            image = img_b64decoded


        elif image is None:
            raise ValueError('Missing image, check image or bucket path.')

        else:
            raise ValueError("Only base 64 encoded image bytes or S3Object are
 supported.")

        response = rekognition.detect_labels(Image={'Bytes': image})
        lambda_response = {
            "statusCode": 200,
            "body": json.dumps(response)
        }
        labels = [label['Name'] for label in response['Labels']]
        print("Labels found:")
        print(labels)

    except ClientError as client_err:

       error_message = "Couldn't analyze image: " + client_err.response['Error']
['Message']

       lambda_response = {
            'statusCode': 400,
            'body': {
                "Error": client_err.response['Error']['Code'],
                "ErrorMessage": error_message
            }
       }
```

```
        logger.error("Error function %s: %s",
                     context.invoked_function_arn, error_message)


    except ValueError as val_error:

        lambda_response = {
            'statusCode': 400,
            'body': {
                "Error": "ValueError",
                "ErrorMessage": format(val_error)
            }
        }
        logger.error("Error function %s: %s",
                     context.invoked_function_arn, format(val_error))

    return lambda_response
```

3. Elija Implementar para implementar la función de Lambda.

Cómo añadir código Python (consola)

Ahora que creó la función de Lambda, puede invocarla para detectar etiquetas en una imagen.

En este paso, ejecute el código Python de su ordenador, que pasa una imagen local, o la imagen de un bucket de Amazon S3, a la función de Lambda.

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 función de Lambda devuelve un error de tiempo de espera, amplíe el período de tiempo de espera de la función de Lambda. Para obtener más información, consulte Configuración del tiempo de espera de la función (consola).

Para obtener más información sobre cómo invocar una función de Lambda mediante el código, consulte Invocación de funciones de Lambda.

Cómo probar la función de Lambda

1. Si aún no lo ha hecho, haga lo siguiente:

a. Asegúrese de que el usuario tiene el permiso `lambda:InvokeFunction`. Puede utilizar la siguiente política:

   Puede obtener el ARN de la función de Lambda en la descripción general de la función en la consola de Lambda.

   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 agregue 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 2: Configure y AWS CLI AWS SDKs.

2. Guarde el siguiente código en un archivo denominado `client.py`:

```
import boto3
import json
import base64
import pprint

# Replace with the name of your S3 bucket and image object key
bucket_name = "name of bucket"
object_key = "name of file in s3 bucket"
# If using a local file, supply the file name as the value of image_path below
image_path = ""
```

```
# Create session and establish connection to client['
session = boto3.Session(profile_name='developer-role')
s3 = session.client('s3', region_name="us-east-1")
lambda_client = session.client('lambda',  region_name="us-east-1")

# Replace with the name of your Lambda function
function_name = 'RekDetectLabels'

def analyze_image_local(img_path):

    print("Analyzing local image:")

    with open(img_path, 'rb') as image_file:
        image_bytes = image_file.read()
        data = base64.b64encode(image_bytes).decode("utf8")

        lambda_payload = {"image": data}

        # Invoke the Lambda function with the event payload
        response = lambda_client.invoke(
            FunctionName=function_name,
            Payload=(json.dumps(lambda_payload))
        )

        decoded = json.loads(response['Payload'].read().decode())
        pprint.pprint(decoded)

def analyze_image_s3(bucket_name, object_key):

    print("Analyzing image in S3 bucket:")

    # Load the image data from S3 into memory
    response = s3.get_object(Bucket=bucket_name, Key=object_key)
    image_data = response['Body'].read()
    image_data = base64.b64encode(image_data).decode("utf8")

     # Create the Lambda event payload
    event = {
        'S3Bucket': bucket_name,
        'S3Object': object_key,
        'ImageBytes': image_data
    }

    # Invoke the Lambda function with the event payload
```

```
    response = lambda_client.invoke(
            FunctionName=function_name,
            InvocationType='RequestResponse',
            Payload=json.dumps(event),
            )

    decoded = json.loads(response['Payload'].read().decode())
    pprint.pprint(decoded)

def main(path_to_image, name_s3_bucket, obj_key):

    if str(path_to_image) != "":
        analyze_image_local(path_to_image)
    else:
        analyze_image_s3(name_s3_bucket, obj_key)

if __name__ == "__main__":
    main(image_path, bucket_name, object_key)
```

3. Ejecute el código. Si el documento está en un bucket de Amazon S3, asegúrese de que es el mismo bucket que indicó anteriormente en el paso 12 de ???.

   Si se ejecuta correctamente, el código devuelve una respuesta JSON parcial para cada tipo de bloque detectado en el documento.

Ejemplos de código para Amazon Rekognition mediante AWS SDKs

Los siguientes ejemplos de código muestran cómo usar Amazon Rekognition AWS con un kit de desarrollo de software (SDK). Los ejemplos de código de este capítulo pretenden complementar los ejemplos de código que se encuentran en el resto de esta guía.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las distintas funciones de servicio, es posible ver las acciones en contexto en los escenarios relacionados.

Los escenarios son ejemplos de código que muestran cómo llevar a cabo una tarea específica a través de llamadas a varias funciones dentro del servicio o combinado con otros Servicios de AWS.

Para obtener una lista completa de guías para desarrolladores del AWS SDK y ejemplos de código, consulte. Uso de Rekognition con un SDK AWS En este tema también se incluye información sobre cómo comenzar a utilizar el SDK y detalles sobre sus versiones anteriores.

Ejemplos de código

- Ejemplos básicos de Amazon Rekognition mediante AWS SDKs
  - Introducción a Amazon Rekognition
  - Acciones para Amazon Rekognition mediante AWS SDKs
    - Úselo CompareFaces con un AWS SDK o CLI
    - Úselo CreateCollection con un AWS SDK o CLI
    - Úselo DeleteCollection con un AWS SDK o CLI
    - Úselo DeleteFaces con un AWS SDK o CLI
    - Úselo DescribeCollection con un AWS SDK o CLI
    - Úselo DetectFaces con un AWS SDK o CLI
    - Úselo DetectLabels con un AWS SDK o CLI
    - Úselo DetectModerationLabels con un AWS SDK o CLI
    - Úselo DetectText con un AWS SDK o CLI
    - Úselo GetCelebrityInfo con un AWS SDK o CLI
    - Úselo IndexFaces con un AWS SDK o CLI
    - Úselo ListCollections con un AWS SDK o CLI

    - Úselo ListFaces con un AWS SDK o CLI
    - Úselo RecognizeCelebrities con un AWS SDK o CLI
    - Úselo SearchFaces con un AWS SDK o CLI
    - Úselo SearchFacesByImage con un AWS SDK o CLI
- Escenarios para el uso de Amazon Rekognition AWS SDKs
    - Crea una colección de Amazon Rekognition y encuentra rostros en ella con un SDK AWS
    - Creación de una aplicación de administración de activos fotográficos que permita a los usuarios administrar las fotos mediante etiquetas
    - Detecte el PPE en las imágenes con Amazon Rekognition AWS mediante un SDK
    - Detecte y muestre elementos en imágenes con Amazon Rekognition AWS mediante un SDK
    - Detecta rostros en una imagen mediante un AWS SDK
    - Detecte información en vídeos con Amazon Rekognition y el SDK AWS
    - Detecte objetos en imágenes con Amazon Rekognition AWS mediante un SDK
    - Detecte personas y objetos en un vídeo con Amazon Rekognition AWS mediante un SDK
    - Guarde el EXIF y otra información de imagen mediante un SDK AWS

Ejemplos básicos de Amazon Rekognition mediante AWS SDKs

Los siguientes ejemplos de código muestran cómo utilizar los conceptos básicos de Amazon Rekognition con. AWS SDKs

Ejemplos

- Introducción a Amazon Rekognition
- Acciones para Amazon Rekognition mediante AWS SDKs
    - Úselo CompareFaces con un AWS SDK o CLI
    - Úselo CreateCollection con un AWS SDK o CLI
    - Úselo DeleteCollection con un AWS SDK o CLI
    - Úselo DeleteFaces con un AWS SDK o CLI
    - Úselo DescribeCollection con un AWS SDK o CLI
    - Úselo DetectFaces con un AWS SDK o CLI
    - Úselo DetectLabels con un AWS SDK o CLI
    - Úselo DetectModerationLabels con un AWS SDK o CLI

- Úselo DetectText con un AWS SDK o CLI
- Úselo GetCelebrityInfo con un AWS SDK o CLI
- Úselo IndexFaces con un AWS SDK o CLI
- Úselo ListCollections con un AWS SDK o CLI
- Úselo ListFaces con un AWS SDK o CLI
- Úselo RecognizeCelebrities con un AWS SDK o CLI
- Úselo SearchFaces con un AWS SDK o CLI
- Úselo SearchFacesByImage con un AWS SDK o CLI

Introducción a Amazon Rekognition

En el siguiente ejemplo de código se muestra cómo empezar a utilizar Amazon Rekognition.

C++

SDK para C++

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

Código para el CMake archivo CMake Lists.txt.

```
# Set the minimum required version of CMake for this project.
cmake_minimum_required(VERSION 3.13)

# Set the AWS service components used by this project.
set(SERVICE_COMPONENTS rekognition)

# Set this project's name.
project("hello_rekognition")

# Set the C++ standard to use to build this target.
# At least C++ 11 is required for the AWS SDK for C++.
set(CMAKE_CXX_STANDARD 11)
```

```
# Use the MSVC variable to determine if this is a Windows build.
set(WINDOWS_BUILD ${MSVC})

if (WINDOWS_BUILD) # Set the location where CMake can find the installed
 libraries for the AWS SDK.
    string(REPLACE ";" "/aws-cpp-sdk-all;" SYSTEM_MODULE_PATH
 "${CMAKE_SYSTEM_PREFIX_PATH}/aws-cpp-sdk-all")
    list(APPEND CMAKE_PREFIX_PATH ${SYSTEM_MODULE_PATH})
endif ()

# Find the AWS SDK for C++ package.
find_package(AWSSDK REQUIRED COMPONENTS ${SERVICE_COMPONENTS})

if (WINDOWS_BUILD AND AWSSDK_INSTALL_AS_SHARED_LIBS)
     # Copy relevant AWS SDK for C++ libraries into the current binary directory
 for running and debugging.

     # set(BIN_SUB_DIR "/Debug") # If you are building from the command line, you
 may need to uncomment this
                                    # and set the proper subdirectory to the
 executables' location.

     AWSSDK_CPY_DYN_LIBS(SERVICE_COMPONENTS ""
 ${CMAKE_CURRENT_BINARY_DIR}${BIN_SUB_DIR})
endif ()

add_executable(${PROJECT_NAME}
        hello_rekognition.cpp)

target_link_libraries(${PROJECT_NAME}
        ${AWSSDK_LINK_LIBRARIES})
```

Código del archivo de origen hello_rekognition.cpp.

```
#include <aws/core/Aws.h>
#include <aws/rekognition/RekognitionClient.h>
#include <aws/rekognition/model/ListCollectionsRequest.h>
#include <iostream>

/*
 *  A "Hello Rekognition" starter application which initializes an Amazon
 Rekognition client and
```

```
 *  lists the Amazon Rekognition collections in the current account and region.
 *
 *  main function
 *
 *  Usage: 'hello_rekognition'
 *
 */

int main(int argc, char **argv) {
    Aws::SDKOptions options;
    //  Optional: change the log level for debugging.
    //  options.loggingOptions.logLevel = Aws::Utils::Logging::LogLevel::Debug;
    Aws::InitAPI(options); // Should only be called once.
    {
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

        Aws::Rekognition::RekognitionClient rekognitionClient(clientConfig);
        Aws::Rekognition::Model::ListCollectionsRequest request;
        Aws::Rekognition::Model::ListCollectionsOutcome outcome =
                rekognitionClient.ListCollections(request);

        if (outcome.IsSuccess()) {
            const Aws::Vector<Aws::String>& collectionsIds =
 outcome.GetResult().GetCollectionIds();
            if (!collectionsIds.empty()) {
                std::cout << "collectionsIds: " << std::endl;
                for (auto &collectionId : collectionsIds) {
                    std::cout << "- " << collectionId << std::endl;
                }
            } else {
                std::cout << "No collections found" << std::endl;
            }
        } else {
            std::cerr << "Error with ListCollections: " << outcome.GetError()
                      << std::endl;
        }
    }


    Aws::ShutdownAPI(options); // Should only be called once.
    return 0;
}
```

- Para obtener más información sobre la API, consulte ListCollectionsla Referencia AWS SDK para C++ de la API.

Para obtener una lista completa de guías para desarrolladores del AWS SDK y ejemplos de código, consulteUso de Rekognition con un SDK AWS. En este tema también se incluye información sobre cómo comenzar a utilizar el SDK y detalles sobre sus versiones anteriores.

Acciones para Amazon Rekognition mediante AWS SDKs

Los siguientes ejemplos de código muestran cómo realizar acciones individuales de Amazon Rekognition con. AWS SDKs Cada ejemplo incluye un enlace a GitHub, donde puede encontrar instrucciones para configurar y ejecutar el código.

Estos fragmentos llaman a la API de Amazon Rekognition y son fragmentos de código de programas más grandes que se deben ejecutar en contexto. Puede ver las acciones en contexto en Escenarios para el uso de Amazon Rekognition AWS SDKs.

Los siguientes ejemplos incluyen solo las acciones que se utilizan con mayor frecuencia. Para ver una lista completa, consulte la Referencia de la API de Amazon Rekognition.

Ejemplos

- Úselo CompareFaces con un AWS SDK o CLI
- Úselo CreateCollection con un AWS SDK o CLI
- Úselo DeleteCollection con un AWS SDK o CLI
- Úselo DeleteFaces con un AWS SDK o CLI
- Úselo DescribeCollection con un AWS SDK o CLI
- Úselo DetectFaces con un AWS SDK o CLI
- Úselo DetectLabels con un AWS SDK o CLI
- Úselo DetectModerationLabels con un AWS SDK o CLI
- Úselo DetectText con un AWS SDK o CLI
- Úselo GetCelebrityInfo con un AWS SDK o CLI
- Úselo IndexFaces con un AWS SDK o CLI
- Úselo ListCollections con un AWS SDK o CLI
- Úselo ListFaces con un AWS SDK o CLI

- Úselo RecognizeCelebrities con un AWS SDK o CLI
- Úselo SearchFaces con un AWS SDK o CLI
- Úselo SearchFacesByImage con un AWS SDK o CLI

Úselo **`CompareFaces`** con un AWS SDK o CLI

Los siguientes ejemplos de código muestran cómo utilizar `CompareFaces`.

Para obtener más información, consulte Comparación de rostros en imágenes.

.NET

SDK para .NET

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
    using System;
    using System.IO;
    using System.Threading.Tasks;
    using Amazon.Rekognition;
    using Amazon.Rekognition.Model;

    /// <summary>
    /// Uses the Amazon Rekognition Service to compare faces in two images.
    /// </summary>
    public class CompareFaces
    {
        public static async Task Main()
        {
            float similarityThreshold = 70F;
            string sourceImage = "source.jpg";
            string targetImage = "target.jpg";

            var rekognitionClient = new AmazonRekognitionClient();

            Amazon.Rekognition.Model.Image imageSource = new
Amazon.Rekognition.Model.Image();
```

```
            try
            {
                using FileStream fs = new FileStream(sourceImage, FileMode.Open,
FileAccess.Read);
                byte[] data = new byte[fs.Length];
                fs.Read(data, 0, (int)fs.Length);
                imageSource.Bytes = new MemoryStream(data);
            }
            catch (Exception)
            {
                Console.WriteLine($"Failed to load source image: {sourceImage}");
                return;
            }

            Amazon.Rekognition.Model.Image imageTarget = new
Amazon.Rekognition.Model.Image();

            try
            {
                using FileStream fs = new FileStream(targetImage, FileMode.Open,
FileAccess.Read);
                byte[] data = new byte[fs.Length];
                data = new byte[fs.Length];
                fs.Read(data, 0, (int)fs.Length);
                imageTarget.Bytes = new MemoryStream(data);
            }
            catch (Exception ex)
            {
                Console.WriteLine($"Failed to load target image: {targetImage}");
                Console.WriteLine(ex.Message);
                return;
            }

            var compareFacesRequest = new CompareFacesRequest
            {
                SourceImage = imageSource,
                TargetImage = imageTarget,
                SimilarityThreshold = similarityThreshold,
            };

            // Call operation
            var compareFacesResponse = await
rekognitionClient.CompareFacesAsync(compareFacesRequest);
```

```
            // Display results
            compareFacesResponse.FaceMatches.ForEach(match =>
            {
                ComparedFace face = match.Face;
                BoundingBox position = face.BoundingBox;
                Console.WriteLine($"Face at {position.Left} {position.Top}
matches with {match.Similarity}% confidence.");
            });

            Console.WriteLine($"Found {compareFacesResponse.UnmatchedFaces.Count}
face(s) that did not match.");
        }
    }
```

- Para obtener más información sobre la API, consulta CompareFaceslla Referencia AWS SDK para .NET de la API.

CLI

AWS CLI

Comparación de rostros en dos imágenes

El siguiente comando `compare-faces` compara rostros en dos imágenes almacenadas en un bucket de Amazon S3.

```
aws rekognition compare-faces \
    --source-image '{"S3Object":
{"Bucket":"MyImageS3Bucket","Name":"source.jpg"}}' \
    --target-image '{"S3Object":
{"Bucket":"MyImageS3Bucket","Name":"target.jpg"}}'
```

Salida:

```
{
    "UnmatchedFaces": [],
    "FaceMatches": [
        {
            "Face": {
```

```
                "BoundingBox": {
                    "Width": 0.12368916720151901,
                    "Top": 0.16007372736930847,
                    "Left": 0.5901257991790771,
                    "Height": 0.25140416622161865
                },
                "Confidence": 100.0,
                "Pose": {
                    "Yaw": -3.7351467609405518,
                    "Roll": -0.10309021919965744,
                    "Pitch": 0.8637830018997192
                },
                "Quality": {
                    "Sharpness": 95.51618957519531,
                    "Brightness": 65.29893493652344
                },
                "Landmarks": [
                    {
                        "Y": 0.26721030473709106,
                        "X": 0.6204193830490112,
                        "Type": "eyeLeft"
                    },
                    {
                        "Y": 0.26831310987472534,
                        "X": 0.6776827573776245,
                        "Type": "eyeRight"
                    },
                    {
                        "Y": 0.3514654338359833,
                        "X": 0.6241428852081299,
                        "Type": "mouthLeft"
                    },
                    {
                        "Y": 0.35258132219314575,
                        "X": 0.6713621020317078,
                        "Type": "mouthRight"
                    },
                    {
                        "Y": 0.3140771687030792,
                        "X": 0.6428444981575012,
                        "Type": "nose"
                    }
                ]
            },
```

```
            "Similarity": 100.0
        }
    ],
    "SourceImageFace": {
        "BoundingBox": {
            "Width": 0.12368916720151901,
            "Top": 0.16007372736930847,
            "Left": 0.5901257991790771,
            "Height": 0.25140416622161865
        },
        "Confidence": 100.0
    }
}
```

Para obtener más información, consulte Comparación de rostros en imágenes en la Guía para desarrolladores de Amazon Rekognition.

- Para obtener más información sobre la API, consulta CompareFaceslа Referencia de AWS CLI comandos.

Java

SDK para Java 2.x

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.*;
import software.amazon.awssdk.core.SdkBytes;

import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;
import java.util.List;

/**
```

```
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class CompareFaces {
    public static void main(String[] args) {
        final String usage = """
            Usage: <bucketName> <sourceKey> <targetKey>

            Where:
                bucketName - The name of the S3 bucket where the images are
 stored.
                sourceKey  - The S3 key (file name) for the source image.
                targetKey  - The S3 key (file name) for the target image.
           """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String sourceKey = args[1];
        String targetKey = args[2];

        Region region = Region.US_WEST_2;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();
        compareTwoFaces(rekClient, bucketName, sourceKey, targetKey);
    }

   /**
    * Compares two faces from images stored in an Amazon S3 bucket using AWS
 Rekognition.
    *
    * <p>This method takes two image keys from an S3 bucket and compares the
 faces within them.
    * It prints out the confidence level of matched faces and reports the number
 of unmatched faces.</p>
```

```
    *
    * @param rekClient   The {@link RekognitionClient} used to call AWS
Rekognition.
    * @param bucketName  The name of the S3 bucket containing the images.
    * @param sourceKey   The object key (file path) for the source image in the
S3 bucket.
    * @param targetKey   The object key (file path) for the target image in the
S3 bucket.
    * @throws RuntimeException If the Rekognition service returns an error.
    */
   public static void compareTwoFaces(RekognitionClient rekClient, String
bucketName, String sourceKey, String targetKey) {
       try {
           Float similarityThreshold = 70F;
           S3Object s3ObjectSource = S3Object.builder()
                   .bucket(bucketName)
                   .name(sourceKey)
                   .build();

           Image sourceImage = Image.builder()
                   .s3Object(s3ObjectSource)
                   .build();

           S3Object s3ObjectTarget = S3Object.builder()
                   .bucket(bucketName)
                   .name(targetKey)
                   .build();

           Image targetImage = Image.builder()
                   .s3Object(s3ObjectTarget)
                   .build();

           CompareFacesRequest facesRequest = CompareFacesRequest.builder()
                   .sourceImage(sourceImage)
                   .targetImage(targetImage)
                   .similarityThreshold(similarityThreshold)
                   .build();

           // Compare the two images.
           CompareFacesResponse compareFacesResult =
rekClient.compareFaces(facesRequest);
           List<CompareFacesMatch> faceDetails =
compareFacesResult.faceMatches();
```

```
            for (CompareFacesMatch match : faceDetails) {
                ComparedFace face = match.face();
                BoundingBox position = face.boundingBox();
                System.out.println("Face at " + position.left().toString()
                        + " " + position.top()
                        + " matches with " + face.confidence().toString()
                        + "% confidence.");
            }

            List<ComparedFace> unmatchedFaces =
 compareFacesResult.unmatchedFaces();
            System.out.println("There were " + unmatchedFaces.size() + " face(s)
 that did not match.");

        } catch (RekognitionException e) {
            System.err.println("Error comparing faces: " +
 e.awsErrorDetails().errorMessage());
            throw new RuntimeException(e);
        }
    }
}
```

- Para obtener más información sobre la API, consulta CompareFaceslla Referencia AWS SDK for Java 2.x de la API.

Kotlin

SDK para Kotlin

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun compareTwoFaces(
    similarityThresholdVal: Float,
    sourceImageVal: String,
    targetImageVal: String,
) {
```

```
    val sourceBytes = (File(sourceImageVal).readBytes())
    val targetBytes = (File(targetImageVal).readBytes())

    // Create an Image object for the source image.
    val souImage =
        Image {
            bytes = sourceBytes
        }

    val tarImage =
        Image {
            bytes = targetBytes
        }

    val facesRequest =
        CompareFacesRequest {
            sourceImage = souImage
            targetImage = tarImage
            similarityThreshold = similarityThresholdVal
        }

    RekognitionClient.fromEnvironment { region = "us-east-1" }.use { rekClient ->

        val compareFacesResult = rekClient.compareFaces(facesRequest)
        val faceDetails = compareFacesResult.faceMatches

        if (faceDetails != null) {
            for (match: CompareFacesMatch in faceDetails) {
                val face = match.face
                val position = face?.boundingBox
                if (position != null) {
                    println("Face at ${position.left} ${position.top} matches
with ${face.confidence} % confidence.")
                }
            }
        }

        val uncompared = compareFacesResult.unmatchedFaces
        if (uncompared != null) {
            println("There was ${uncompared.size} face(s) that did not match")
        }

        println("Source image rotation:
${compareFacesResult.sourceImageOrientationCorrection}")
```

```
        println("target image rotation:
 ${compareFacesResult.targetImageOrientationCorrection}")
    }
}
```

- Para obtener más información sobre la API, consulta CompareFaceslla referencia sobre el AWS SDK para la API de Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
class RekognitionImage:
    """
    Encapsulates an Amazon Rekognition image. This class is a thin wrapper
    around parts of the Boto3 Amazon Rekognition API.
    """

    def __init__(self, image, image_name, rekognition_client):
        """
        Initializes the image object.

        :param image: Data that defines the image, either the image bytes or
                      an Amazon S3 bucket and object key.
        :param image_name: The name of the image.
        :param rekognition_client: A Boto3 Rekognition client.
        """
        self.image = image
        self.image_name = image_name
        self.rekognition_client = rekognition_client


    def compare_faces(self, target_image, similarity):
        """
```

```
        Compares faces in the image with the largest face in the target image.

        :param target_image: The target image to compare against.
        :param similarity: Faces in the image must have a similarity value
greater
                           than this value to be included in the results.
        :return: A tuple. The first element is the list of faces that match the
                 reference image. The second element is the list of faces that
have
                 a similarity value below the specified threshold.
        """
        try:
            response = self.rekognition_client.compare_faces(
                SourceImage=self.image,
                TargetImage=target_image.image,
                SimilarityThreshold=similarity,
            )
            matches = [
                RekognitionFace(match["Face"]) for match in
response["FaceMatches"]
            ]
            unmatches = [RekognitionFace(face) for face in
response["UnmatchedFaces"]]
            logger.info(
                "Found %s matched faces and %s unmatched faces.",
                len(matches),
                len(unmatches),
            )
        except ClientError:
            logger.exception(
                "Couldn't match faces from %s to %s.",
                self.image_name,
                target_image.image_name,
            )
            raise
        else:
            return matches, unmatches
```

- Para obtener más información sobre la API, consulta CompareFaces la AWS Referencia de API de SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
TRY.
    " Create S3 object reference for the source image
    DATA(lo_source_s3obj) = NEW /aws1/cl_reks3object(
      iv_bucket = iv_source_s3_bucket
      iv_name = iv_source_s3_key ).

    " Create source image object
    DATA(lo_source_image) = NEW /aws1/cl_rekimage(
      io_s3object = lo_source_s3obj ).

    " Create S3 object reference for the target image
    DATA(lo_target_s3obj) = NEW /aws1/cl_reks3object(
      iv_bucket = iv_target_s3_bucket
      iv_name = iv_target_s3_key ).

    " Create target image object
    DATA(lo_target_image) = NEW /aws1/cl_rekimage(
      io_s3object = lo_target_s3obj ).

    " Compare faces
    oo_result = lo_rek->comparefaces(
      io_sourceimage = lo_source_image
      io_targetimage = lo_target_image
      iv_similaritythreshold = iv_similarity ).

    DATA(lt_face_matches) = oo_result->get_facematches( ).
    DATA(lt_unmatched_faces) = oo_result->get_unmatchedfaces( ).

    " Get counts of matched and unmatched faces
    DATA(lv_matched_count) = lines( lt_face_matches ).
    DATA(lv_unmatched_count) = lines( lt_unmatched_faces ).
```

```
      " Output detailed comparison results
      DATA(lv_message) = |Face comparison completed: | &&
                         |{ lv_matched_count } matched face(s), | &&
                         |{ lv_unmatched_count } unmatched face(s).|.
      MESSAGE lv_message TYPE 'I'.
    CATCH /aws1/cx_rekinvalids3objectex.
      MESSAGE 'Invalid S3 object.' TYPE 'E'.
    CATCH /aws1/cx_rekinvalidparameterex.
      MESSAGE 'Invalid parameter value.' TYPE 'E'.
  ENDTRY.
```

- Para obtener más información sobre la API, consulte CompareFaceslа referencia sobre la API ABAP del AWS SDK para SAP.

Para obtener una lista completa de guías para desarrolladores del AWS SDK y ejemplos de código, consulteUso de Rekognition con un SDK AWS. En este tema también se incluye información sobre cómo comenzar a utilizar el SDK y detalles sobre sus versiones anteriores.

Úselo **`CreateCollection`** con un AWS SDK o CLI

Los siguientes ejemplos de código muestran cómo utilizar `CreateCollection`.

Para obtener más información, consulte Creación de una colección.

.NET

SDK para .NET

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
using System;
using System.Threading.Tasks;
using Amazon.Rekognition;
using Amazon.Rekognition.Model;
```

```
    /// <summary>
    /// Uses Amazon Rekognition to create a collection to which you can add
    /// faces using the IndexFaces operation.
    /// </summary>
    public class CreateCollection
    {
        public static async Task Main()
        {
            var rekognitionClient = new AmazonRekognitionClient();

            string collectionId = "MyCollection";
            Console.WriteLine("Creating collection: " + collectionId);

            var createCollectionRequest = new CreateCollectionRequest
            {
                CollectionId = collectionId,
            };

            CreateCollectionResponse createCollectionResponse = await
rekognitionClient.CreateCollectionAsync(createCollectionRequest);
            Console.WriteLine($"CollectionArn :
{createCollectionResponse.CollectionArn}");
            Console.WriteLine($"Status code :
{createCollectionResponse.StatusCode}");
        }
    }
```

- Para obtener más información sobre la API, consulta CreateCollectionla Referencia AWS SDK para .NET de la API.

CLI

AWS CLI

Creación de una colección

El siguiente comando `create-collection` crea una colección con el nombre especificado.

```
aws rekognition create-collection \
    --collection-id "MyCollection"
```

Salida:

```
{
    "CollectionArn": "aws:rekognition:us-west-2:123456789012:collection/
MyCollection",
    "FaceModelVersion": "4.0",
    "StatusCode": 200
}
```

Para obtener más información, consulte Creación de una colección en la Guía para desarrolladores de Amazon Rekognition.

- Para obtener más información sobre la API, consulta CreateCollectionla Referencia de AWS CLI comandos.

Java

SDK para Java 2.x

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import
 software.amazon.awssdk.services.rekognition.model.CreateCollectionResponse;
import software.amazon.awssdk.services.rekognition.model.CreateCollectionRequest;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
```

```
public class CreateCollection {
    public static void main(String[] args) {
        final String usage = """

            Usage: <collectionName>\s

            Where:
                collectionName - The name of the collection.\s
            """;

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

        String collectionId = args[0];
        Region region = Region.US_WEST_2;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        System.out.println("Creating collection: " + collectionId);
        createMyCollection(rekClient, collectionId);
        rekClient.close();
    }

    /**
     * Creates a new Amazon Rekognition collection.
     *
     * @param rekClient    the Amazon Rekognition client used to interact with
 the Rekognition service
     * @param collectionId the unique identifier for the collection to be created
     */
    public static void createMyCollection(RekognitionClient rekClient, String
 collectionId) {
        try {
            CreateCollectionRequest collectionRequest =
 CreateCollectionRequest.builder()
                    .collectionId(collectionId)
                    .build();

            CreateCollectionResponse collectionResponse =
 rekClient.createCollection(collectionRequest);
```

```
            System.out.println("CollectionArn: " +
 collectionResponse.collectionArn());
            System.out.println("Status code: " +
 collectionResponse.statusCode().toString());

        } catch (RekognitionException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Para obtener más información sobre la API, consulta CreateCollectionla Referencia AWS SDK for Java 2.x de la API.

Kotlin

SDK para Kotlin

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun createMyCollection(collectionIdVal: String) {
    val request =
        CreateCollectionRequest {
            collectionId = collectionIdVal
        }

    RekognitionClient.fromEnvironment { region = "us-east-1" }.use { rekClient ->
        val response = rekClient.createCollection(request)
        println("Collection ARN is ${response.collectionArn}")
        println("Status code is ${response.statusCode}")
    }
}
```

- Para obtener más información sobre la API, consulta CreateCollectionla referencia sobre el AWS SDK para la API de Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
class RekognitionCollectionManager:
    """
    Encapsulates Amazon Rekognition collection management functions.
    This class is a thin wrapper around parts of the Boto3 Amazon Rekognition
 API.
    """

    def __init__(self, rekognition_client):
        """
        Initializes the collection manager object.

        :param rekognition_client: A Boto3 Rekognition client.
        """
        self.rekognition_client = rekognition_client


    def create_collection(self, collection_id):
        """
        Creates an empty collection.

        :param collection_id: Text that identifies the collection.
        :return: The newly created collection.
        """
        try:
            response = self.rekognition_client.create_collection(
                CollectionId=collection_id
            )
            response["CollectionId"] = collection_id
```

```
            collection = RekognitionCollection(response, self.rekognition_client)
            logger.info("Created collection %s.", collection_id)
        except ClientError:
            logger.exception("Couldn't create collection %s.", collection_id)
            raise
        else:
            return collection
```

- Para obtener más información sobre la API, consulta CreateCollectionla AWS Referencia de API de SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
TRY.
    oo_result = lo_rek->createcollection(
      iv_collectionid = iv_collection_id ).
    MESSAGE 'Collection created successfully.' TYPE 'I'.
  CATCH /aws1/cx_rekresrcalrdyexistsex.
    MESSAGE 'Collection already exists.' TYPE 'E'.
  CATCH /aws1/cx_rekinvalidparameterex.
    MESSAGE 'Invalid parameter value.' TYPE 'E'.
ENDTRY.
```

- Para obtener más información sobre la API, consulte CreateCollectionla referencia sobre la API ABAP del AWS SDK para SAP.

Para obtener una lista completa de guías para desarrolladores del AWS SDK y ejemplos de código, consulteUso de Rekognition con un SDK AWS. En este tema también se incluye información sobre cómo comenzar a utilizar el SDK y detalles sobre sus versiones anteriores.

Úselo **`DeleteCollection`** con un AWS SDK o CLI

Los siguientes ejemplos de código muestran cómo utilizar `DeleteCollection`.

Para obtener más información, consulte Eliminación de una colección.

.NET

SDK para .NET

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
using System;
using System.Threading.Tasks;
using Amazon.Rekognition;
using Amazon.Rekognition.Model;

/// <summary>
/// Uses the Amazon Rekognition Service to delete an existing collection.
/// </summary>
public class DeleteCollection
{
    public static async Task Main()
    {
        var rekognitionClient = new AmazonRekognitionClient();

        string collectionId = "MyCollection";
        Console.WriteLine("Deleting collection: " + collectionId);

        var deleteCollectionRequest = new DeleteCollectionRequest()
        {
            CollectionId = collectionId,
        };

        var deleteCollectionResponse = await
rekognitionClient.DeleteCollectionAsync(deleteCollectionRequest);
        Console.WriteLine($"{collectionId}:
{deleteCollectionResponse.StatusCode}");
```

```
        }
    }
```

- Para obtener más información sobre la API, consulta DeleteCollectionla Referencia AWS SDK para .NET de la API.

CLI

AWS CLI

Eliminación de una colección

El siguiente comando `delete-collection` elimina la colección especificada.

```
aws rekognition delete-collection \
    --collection-id MyCollection
```

Salida:

```
{
    "StatusCode": 200
}
```

Para obtener más información, consulte Eliminación de una colección en la Guía para desarrolladores de Amazon Rekognition.

- Para obtener más información sobre la API, consulta DeleteCollectionla Referencia de AWS CLI comandos.

Java

SDK para Java 2.x

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.DeleteCollectionRequest;
import
 software.amazon.awssdk.services.rekognition.model.DeleteCollectionResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DeleteCollection {
    public static void main(String[] args) {
        final String usage = """
            Usage: <collectionId>\s

            Where:
                collectionId - The id of the collection to delete.\s
            """;

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

        String collectionId = args[0];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        System.out.println("Deleting collection: " + collectionId);
        deleteMyCollection(rekClient, collectionId);
        rekClient.close();
    }

    /**
     * Deletes an Amazon Rekognition collection.
```

```
     *
     * @param rekClient      An instance of the {@link RekognitionClient} class,
 which is used to interact with the Amazon Rekognition service.
     * @param collectionId   The ID of the collection to be deleted.
     */
    public static void deleteMyCollection(RekognitionClient rekClient, String
 collectionId) {
        try {
            DeleteCollectionRequest deleteCollectionRequest =
 DeleteCollectionRequest.builder()
                    .collectionId(collectionId)
                    .build();

            DeleteCollectionResponse deleteCollectionResponse =
 rekClient.deleteCollection(deleteCollectionRequest);
            System.out.println(collectionId + ": " +
 deleteCollectionResponse.statusCode().toString());

        } catch (RekognitionException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Para obtener más información sobre la API, consulta DeleteCollectionla Referencia AWS SDK for Java 2.x de la API.

Kotlin

SDK para Kotlin

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun deleteMyCollection(collectionIdVal: String) {
    val request =
```

```
        DeleteCollectionRequest {
            collectionId = collectionIdVal
        }

    RekognitionClient.fromEnvironment { region = "us-east-1" }.use { rekClient ->
        val response = rekClient.deleteCollection(request)
        println("The collectionId status is ${response.statusCode}")
    }
}
```

- Para obtener más información sobre la API, consulta DeleteCollectionla referencia sobre el AWS SDK para la API de Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
class RekognitionCollection:
    """
    Encapsulates an Amazon Rekognition collection. This class is a thin wrapper
    around parts of the Boto3 Amazon Rekognition API.
    """

    def __init__(self, collection, rekognition_client):
        """
        Initializes a collection object.

        :param collection: Collection data in the format returned by a call to
                           create_collection.
        :param rekognition_client: A Boto3 Rekognition client.
        """
        self.collection_id = collection["CollectionId"]
        self.collection_arn, self.face_count, self.created =
 self._unpack_collection(
```

```
            collection
        )
        self.rekognition_client = rekognition_client

    @staticmethod
    def _unpack_collection(collection):
        """
        Unpacks optional parts of a collection that can be returned by
        describe_collection.

        :param collection: The collection data.
        :return: A tuple of the data in the collection.
        """
        return (
            collection.get("CollectionArn"),
            collection.get("FaceCount", 0),
            collection.get("CreationTimestamp"),
        )


    def delete_collection(self):
        """
        Deletes the collection.
        """
        try:

self.rekognition_client.delete_collection(CollectionId=self.collection_id)
            logger.info("Deleted collection %s.", self.collection_id)
            self.collection_id = None
        except ClientError:
            logger.exception("Couldn't delete collection %s.",
self.collection_id)
            raise
```

- Para obtener más información sobre la API, consulta DeleteCollectionla AWS Referencia de API de SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
TRY.
    lo_rek->deletecollection(
      iv_collectionid = iv_collection_id ).
    MESSAGE 'Collection deleted successfully.' TYPE 'I'.
  CATCH /aws1/cx_rekresourcenotfoundex.
    MESSAGE 'Collection not found.' TYPE 'E'.
  CATCH /aws1/cx_rekinvalidparameterex.
    MESSAGE 'Invalid parameter value.' TYPE 'E'.
ENDTRY.
```

- Para obtener más información sobre la API, consulte DeleteCollectionla referencia sobre la API ABAP del AWS SDK para SAP.

Para obtener una lista completa de guías para desarrolladores del AWS SDK y ejemplos de código, consulteUso de Rekognition con un SDK AWS. En este tema también se incluye información sobre cómo comenzar a utilizar el SDK y detalles sobre sus versiones anteriores.

Úselo **`DeleteFaces`** con un AWS SDK o CLI

Los siguientes ejemplos de código muestran cómo utilizar `DeleteFaces`.

Para obtener más información, consulte Eliminación de rostros de una colección.

.NET

SDK para .NET

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
    using System;
    using System.Collections.Generic;
    using System.Threading.Tasks;
    using Amazon.Rekognition;
    using Amazon.Rekognition.Model;

    /// <summary>
    /// Uses the Amazon Rekognition Service to delete one or more faces from
    /// a Rekognition collection.
    /// </summary>
    public class DeleteFaces
    {
        public static async Task Main()
        {
            string collectionId = "MyCollection";
            var faces = new List<string> { "xxxxxxxx-xxxx-xxxx-xxxx-
xxxxxxxxxxxx" };

            var rekognitionClient = new AmazonRekognitionClient();

            var deleteFacesRequest = new DeleteFacesRequest()
            {
                CollectionId = collectionId,
                FaceIds = faces,
            };

            DeleteFacesResponse deleteFacesResponse = await
 rekognitionClient.DeleteFacesAsync(deleteFacesRequest);
            deleteFacesResponse.DeletedFaces.ForEach(face =>
            {
                Console.WriteLine($"FaceID: {face}");
            });
```

```
        }
    }
```

- Para obtener más información sobre la API, consulta DeleteFaceslа Referencia AWS SDK para .NET de la API.

CLI

AWS CLI

Eliminación de los rostros de una colección

El siguiente comando `delete-faces` elimina los rostros especificados de una colección.

```
aws rekognition delete-faces \
    --collection-id MyCollection
    --face-ids '["0040279c-0178-436e-b70a-e61b074e96b0"]'
```

Salida:

```
{
    "DeletedFaces": [
        "0040279c-0178-436e-b70a-e61b074e96b0"
    ]
}
```

Para obtener más información, consulte Eliminación de rostros de una colección en la Guía para desarrolladores de Amazon Rekognition.

- Para obtener más información sobre la API, consulta DeleteFaceslа Referencia de AWS CLI comandos.

Java

SDK para Java 2.x

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.DeleteFacesRequest;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DeleteFacesFromCollection {
    public static void main(String[] args) {
        final String usage = """
            Usage: <collectionId> <faceId>\s

            Where:
                collectionId - The id of the collection from which faces are
 deleted.\s
                faceId - The id of the face to delete.\s
           """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String collectionId = args[0];
        String faceId = args[1];
```

```
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        System.out.println("Deleting collection: " + collectionId);
        deleteFacesCollection(rekClient, collectionId, faceId);
        rekClient.close();
    }

    /**
     * Deletes a face from the specified Amazon Rekognition collection.
     *
     * @param rekClient     an instance of the Amazon Rekognition client
     * @param collectionId  the ID of the collection from which the face should
 be deleted
     * @param faceId        the ID of the face to be deleted
     * @throws RekognitionException if an error occurs while deleting the face
     */
    public static void deleteFacesCollection(RekognitionClient rekClient,
            String collectionId,
            String faceId) {

        try {
            DeleteFacesRequest deleteFacesRequest = DeleteFacesRequest.builder()
                    .collectionId(collectionId)
                    .faceIds(faceId)
                    .build();

            rekClient.deleteFaces(deleteFacesRequest);
            System.out.println("The face was deleted from the collection.");

        } catch (RekognitionException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Para obtener más información sobre la API, consulta DeleteFaces la Referencia AWS SDK for Java 2.x de la API.

Kotlin

SDK para Kotlin

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun deleteFacesCollection(
    collectionIdVal: String?,
    faceIdVal: String,
) {
    val deleteFacesRequest =
        DeleteFacesRequest {
            collectionId = collectionIdVal
            faceIds = listOf(faceIdVal)
        }

    RekognitionClient.fromEnvironment { region = "us-east-1" }.use { rekClient ->
        rekClient.deleteFaces(deleteFacesRequest)
        println("$faceIdVal was deleted from the collection")
    }
}
```

- Para obtener más información sobre la API, consulta DeleteFaceslа referencia sobre el AWS SDK para la API de Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
class RekognitionCollection:
    """
    Encapsulates an Amazon Rekognition collection. This class is a thin wrapper
    around parts of the Boto3 Amazon Rekognition API.
    """

    def __init__(self, collection, rekognition_client):
        """
        Initializes a collection object.

        :param collection: Collection data in the format returned by a call to
                           create_collection.
        :param rekognition_client: A Boto3 Rekognition client.
        """
        self.collection_id = collection["CollectionId"]
        self.collection_arn, self.face_count, self.created =
 self._unpack_collection(
            collection
        )
        self.rekognition_client = rekognition_client

    @staticmethod
    def _unpack_collection(collection):
        """
        Unpacks optional parts of a collection that can be returned by
        describe_collection.

        :param collection: The collection data.
        :return: A tuple of the data in the collection.
        """
        return (
            collection.get("CollectionArn"),
            collection.get("FaceCount", 0),
            collection.get("CreationTimestamp"),
        )


    def delete_faces(self, face_ids):
        """
        Deletes faces from the collection.

        :param face_ids: The list of IDs of faces to delete.
        :return: The list of IDs of faces that were deleted.
```

```
        """
        try:
            response = self.rekognition_client.delete_faces(
                CollectionId=self.collection_id, FaceIds=face_ids
            )
            deleted_ids = response["DeletedFaces"]
            logger.info(
                "Deleted %s faces from %s.", len(deleted_ids), self.collection_id
            )
        except ClientError:
            logger.exception("Couldn't delete faces from %s.",
self.collection_id)
            raise
        else:
            return deleted_ids
```

- Para obtener más información sobre la API, consulta DeleteFaceslala AWS Referencia de API de SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
    TRY.
        oo_result = lo_rek->deletefaces(
          iv_collectionid = iv_collection_id
          it_faceids = it_face_ids ).

        DATA(lt_deleted_faces) = oo_result->get_deletedfaces( ).
        DATA(lv_deleted_count) = lines( lt_deleted_faces ).
        DATA(lv_msg6) = |{ lv_deleted_count } face(s) deleted successfully.|.
        MESSAGE lv_msg6 TYPE 'I'.
```

```
    CATCH /aws1/cx_rekresourcenotfoundex.
      MESSAGE 'Collection not found.' TYPE 'E'.
    CATCH /aws1/cx_rekinvalidparameterex.
      MESSAGE 'Invalid parameter value.' TYPE 'E'.
  ENDTRY.
```

- Para obtener más información sobre la API, consulte DeleteFaceslа referencia sobre la API ABAP del AWS SDK para SAP.

Para obtener una lista completa de guías para desarrolladores del AWS SDK y ejemplos de código, consulteUso de Rekognition con un SDK AWS. En este tema también se incluye información sobre cómo comenzar a utilizar el SDK y detalles sobre sus versiones anteriores.

Úselo **`DescribeCollection`** con un AWS SDK o CLI

Los siguientes ejemplos de código muestran cómo utilizar `DescribeCollection`.

Para obtener más información, consulte Descripción de una colección.

.NET

SDK para .NET

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
using System;
using System.Threading.Tasks;
using Amazon.Rekognition;
using Amazon.Rekognition.Model;

/// <summary>
/// Uses the Amazon Rekognition Service to describe the contents of a
/// collection.
/// </summary>
public class DescribeCollection
```

```
    {
        public static async Task Main()
        {
            var rekognitionClient = new AmazonRekognitionClient();

            string collectionId = "MyCollection";
            Console.WriteLine($"Describing collection: {collectionId}");

            var describeCollectionRequest = new DescribeCollectionRequest()
            {
                CollectionId = collectionId,
            };

            var describeCollectionResponse = await
rekognitionClient.DescribeCollectionAsync(describeCollectionRequest);
            Console.WriteLine($"Collection ARN:
{describeCollectionResponse.CollectionARN}");
            Console.WriteLine($"Face count:
{describeCollectionResponse.FaceCount}");
            Console.WriteLine($"Face model version:
{describeCollectionResponse.FaceModelVersion}");
            Console.WriteLine($"Created:
{describeCollectionResponse.CreationTimestamp}");
        }
    }
```

- Para obtener más información sobre la API, consulta DescribeCollectionla Referencia AWS SDK para .NET de la API.

CLI

AWS CLI

Descripción de una colección

En el siguiente ejemplo de `describe-collection` se muestran los detalles de la colección especificada.

```
aws rekognition describe-collection \
    --collection-id MyCollection
```

Salida:

```
{
    "FaceCount": 200,
    "CreationTimestamp": 1569444828.274,
    "CollectionARN": "arn:aws:rekognition:us-west-2:123456789012:collection/
MyCollection",
    "FaceModelVersion": "4.0"
}
```

Para obtener más información, consulte Descripción de una colección en la Guía para desarrolladores de Amazon Rekognition.

- Para obtener más información sobre la API, consulta DescribeCollectionla Referencia de AWS CLI comandos.

Java

SDK para Java 2.x

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import
 software.amazon.awssdk.services.rekognition.model.DescribeCollectionRequest;
import
 software.amazon.awssdk.services.rekognition.model.DescribeCollectionResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
```

```
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DescribeCollection {
    public static void main(String[] args) {
        final String usage = """
            Usage:    <collectionName>

            Where:
                collectionName - The name of the Amazon Rekognition collection.\s
            """;

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

        String collectionName = args[0];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        describeColl(rekClient, collectionName);
        rekClient.close();
    }

    /**
     * Describes an Amazon Rekognition collection.
     *
     * @param rekClient         The Amazon Rekognition client used to make the
 request.
     * @param collectionName    The name of the collection to describe.
     *
     * @throws RekognitionException If an error occurs while describing the
 collection.
     */
    public static void describeColl(RekognitionClient rekClient, String
 collectionName) {
        try {
            DescribeCollectionRequest describeCollectionRequest =
 DescribeCollectionRequest.builder()
                    .collectionId(collectionName)
                    .build();
```

```
            DescribeCollectionResponse describeCollectionResponse = rekClient
                    .describeCollection(describeCollectionRequest);
            System.out.println("Collection Arn : " +
 describeCollectionResponse.collectionARN());
            System.out.println("Created : " +
 describeCollectionResponse.creationTimestamp().toString());

        } catch (RekognitionException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Para obtener más información sobre la API, consulta DescribeCollectionla Referencia AWS SDK for Java 2.x de la API.

Kotlin

SDK para Kotlin

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun describeColl(collectionName: String) {
    val request =
        DescribeCollectionRequest {
            collectionId = collectionName
        }

    RekognitionClient.fromEnvironment { region = "us-east-1" }.use { rekClient ->
        val response = rekClient.describeCollection(request)
        println("The collection Arn is ${response.collectionArn}")
        println("The collection contains this many faces ${response.faceCount}")
    }
}
```

- Para obtener más información sobre la API, consulta DescribeCollectionla referencia sobre el AWS SDK para la API de Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
class RekognitionCollection:
    """
    Encapsulates an Amazon Rekognition collection. This class is a thin wrapper
    around parts of the Boto3 Amazon Rekognition API.
    """

    def __init__(self, collection, rekognition_client):
        """
        Initializes a collection object.

        :param collection: Collection data in the format returned by a call to
                           create_collection.
        :param rekognition_client: A Boto3 Rekognition client.
        """
        self.collection_id = collection["CollectionId"]
        self.collection_arn, self.face_count, self.created =
 self._unpack_collection(
            collection
        )
        self.rekognition_client = rekognition_client

    @staticmethod
    def _unpack_collection(collection):
        """
        Unpacks optional parts of a collection that can be returned by
        describe_collection.
```

```
        :param collection: The collection data.
        :return: A tuple of the data in the collection.
        """
        return (
            collection.get("CollectionArn"),
            collection.get("FaceCount", 0),
            collection.get("CreationTimestamp"),
        )


    def describe_collection(self):
        """
        Gets data about the collection from the Amazon Rekognition service.

        :return: The collection rendered as a dict.
        """
        try:
            response = self.rekognition_client.describe_collection(
                CollectionId=self.collection_id
            )
            # Work around capitalization of Arn vs. ARN
            response["CollectionArn"] = response.get("CollectionARN")
            (
                self.collection_arn,
                self.face_count,
                self.created,
            ) = self._unpack_collection(response)
            logger.info("Got data for collection %s.", self.collection_id)
        except ClientError:
            logger.exception("Couldn't get data for collection %s.",
self.collection_id)
            raise
        else:
            return self.to_dict()
```

- Para obtener más información sobre la API, consulta DescribeCollectionla AWS Referencia de API de SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
   TRY.
       oo_result = lo_rek->describecollection(
         iv_collectionid = iv_collection_id ).
       DATA(lv_face_count) = oo_result->get_facecount( ).
       DATA(lv_msg) = |Collection described: { lv_face_count } face(s)
indexed.|.
       MESSAGE lv_msg TYPE 'I'.
     CATCH /aws1/cx_rekresourcenotfoundex.
       MESSAGE 'Collection not found.' TYPE 'E'.
     CATCH /aws1/cx_rekinvalidparameterex.
       MESSAGE 'Invalid parameter value.' TYPE 'E'.
   ENDTRY.
```

- Para obtener más información sobre la API, consulte DescribeCollectionla referencia sobre la API ABAP del AWS SDK para SAP.

Para obtener una lista completa de guías para desarrolladores del AWS SDK y ejemplos de código, consulteUso de Rekognition con un SDK AWS. En este tema también se incluye información sobre cómo comenzar a utilizar el SDK y detalles sobre sus versiones anteriores.

Úselo **`DetectFaces`** con un AWS SDK o CLI

Los siguientes ejemplos de código muestran cómo utilizar `DetectFaces`.

Para obtener más información, consulte Detección de rostros en una imagen.

.NET

SDK para .NET

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using Amazon.Rekognition;
using Amazon.Rekognition.Model;

/// <summary>
/// Uses the Amazon Rekognition Service to detect faces within an image
/// stored in an Amazon Simple Storage Service (Amazon S3) bucket.
/// </summary>
public class DetectFaces
{
    public static async Task Main()
    {
        string photo = "input.jpg";
        string bucket = "amzn-s3-demo-bucket";

        var rekognitionClient = new AmazonRekognitionClient();

        var detectFacesRequest = new DetectFacesRequest()
        {
            Image = new Image()
            {
                S3Object = new S3Object()
                {
                    Name = photo,
                    Bucket = bucket,
                },
            },

            // Attributes can be "ALL" or "DEFAULT".
```

```
                // "DEFAULT": BoundingBox, Confidence, Landmarks, Pose, and
 Quality.
                // "ALL": See https://docs.aws.amazon.com/sdkfornet/v3/apidocs/
items/Rekognition/TFaceDetail.html
                Attributes = new List<string>() { "ALL" },
            };

            try
            {
                DetectFacesResponse detectFacesResponse = await
 rekognitionClient.DetectFacesAsync(detectFacesRequest);
                bool hasAll = detectFacesRequest.Attributes.Contains("ALL");
                foreach (FaceDetail face in detectFacesResponse.FaceDetails)
                {
                    Console.WriteLine($"BoundingBox: top={face.BoundingBox.Left}
 left={face.BoundingBox.Top} width={face.BoundingBox.Width}
 height={face.BoundingBox.Height}");
                    Console.WriteLine($"Confidence: {face.Confidence}");
                    Console.WriteLine($"Landmarks: {face.Landmarks.Count}");
                    Console.WriteLine($"Pose: pitch={face.Pose.Pitch}
 roll={face.Pose.Roll} yaw={face.Pose.Yaw}");
                    Console.WriteLine($"Brightness:
 {face.Quality.Brightness}\tSharpness: {face.Quality.Sharpness}");

                    if (hasAll)
                    {
                        Console.WriteLine($"Estimated age is between
 {face.AgeRange.Low} and {face.AgeRange.High} years old.");
                    }
                }
            }
            catch (Exception ex)
            {
                Console.WriteLine(ex.Message);
            }
        }
    }
```

Visualizar información del cuadro delimitador de todos los rostros en una imagen.

```
    using System;
```

```
    using System.Collections.Generic;
    using System.Drawing;
    using System.IO;
    using System.Threading.Tasks;
    using Amazon.Rekognition;
    using Amazon.Rekognition.Model;

    /// <summary>
    /// Uses the Amazon Rekognition Service to display the details of the
    /// bounding boxes around the faces detected in an image.
    /// </summary>
    public class ImageOrientationBoundingBox
    {
        public static async Task Main()
        {
            string photo = @"D:\Development\AWS-Examples\Rekognition
\target.jpg"; // "photo.jpg";

            var rekognitionClient = new AmazonRekognitionClient();

            var image = new Amazon.Rekognition.Model.Image();
            try
            {
                using var fs = new FileStream(photo, FileMode.Open,
 FileAccess.Read);
                byte[] data = null;
                data = new byte[fs.Length];
                fs.Read(data, 0, (int)fs.Length);
                image.Bytes = new MemoryStream(data);
            }
            catch (Exception)
            {
                Console.WriteLine("Failed to load file " + photo);
                return;
            }

            int height;
            int width;

            // Used to extract original photo width/height
            using (var imageBitmap = new Bitmap(photo))
            {
                height = imageBitmap.Height;
                width = imageBitmap.Width;
```

```
            }

            Console.WriteLine("Image Information:");
            Console.WriteLine(photo);
            Console.WriteLine("Image Height: " + height);
            Console.WriteLine("Image Width: " + width);

            try
            {
                var detectFacesRequest = new DetectFacesRequest()
                {
                    Image = image,
                    Attributes = new List<string>() { "ALL" },
                };

                DetectFacesResponse detectFacesResponse = await
rekognitionClient.DetectFacesAsync(detectFacesRequest);
                detectFacesResponse.FaceDetails.ForEach(face =>
                {
                    Console.WriteLine("Face:");
                    ShowBoundingBoxPositions(
                        height,
                        width,
                        face.BoundingBox,
                        detectFacesResponse.OrientationCorrection);

                    Console.WriteLine($"BoundingBox: top={face.BoundingBox.Left}
left={face.BoundingBox.Top} width={face.BoundingBox.Width}
height={face.BoundingBox.Height}");
                    Console.WriteLine($"The detected face is estimated to be
between {face.AgeRange.Low} and {face.AgeRange.High} years old.\n");
                });
            }
            catch (Exception ex)
            {
                Console.WriteLine(ex.Message);
            }
        }

        /// <summary>
        /// Display the bounding box information for an image.
        /// </summary>
        /// <param name="imageHeight">The height of the image.</param>
        /// <param name="imageWidth">The width of the image.</param>
```

```
        /// <param name="box">The bounding box for a face found within the
 image.</param>
        /// <param name="rotation">The rotation of the face's bounding box.</
param>
        public static void ShowBoundingBoxPositions(int imageHeight, int
 imageWidth, BoundingBox box, string rotation)
        {
            float left;
            float top;

            if (rotation == null)
            {
                Console.WriteLine("No estimated orientation. Check Exif data.");
                return;
            }

            // Calculate face position based on image orientation.
            switch (rotation)
            {
                case "ROTATE_0":
                    left = imageWidth * box.Left;
                    top = imageHeight * box.Top;
                    break;
                case "ROTATE_90":
                    left = imageHeight * (1 - (box.Top + box.Height));
                    top = imageWidth * box.Left;
                    break;
                case "ROTATE_180":
                    left = imageWidth - (imageWidth * (box.Left + box.Width));
                    top = imageHeight * (1 - (box.Top + box.Height));
                    break;
                case "ROTATE_270":
                    left = imageHeight * box.Top;
                    top = imageWidth * (1 - box.Left - box.Width);
                    break;
                default:
                    Console.WriteLine("No estimated orientation information.
 Check Exif data.");
                    return;
            }

            // Display face location information.
            Console.WriteLine($"Left: {left}");
            Console.WriteLine($"Top: {top}");
```

```
            Console.WriteLine($"Face Width: {imageWidth * box.Width}");
            Console.WriteLine($"Face Height: {imageHeight * box.Height}");
        }
    }
```

- Para obtener más información sobre la API, consulta DetectFaceslla Referencia AWS SDK para .NET de la API.

CLI

AWS CLI

Detección de rostros en una imagen

El siguiente comando `detect-faces` detecta rostros en la imagen especificada almacenada en un bucket de Amazon S3.

```
aws rekognition detect-faces \
    --image '{"S3Object":{"Bucket":"MyImageS3Bucket","Name":"MyFriend.jpg"}}' \
    --attributes "ALL"
```

Salida:

```
{
    "FaceDetails": [
        {
            "Confidence": 100.0,
            "Eyeglasses": {
                "Confidence": 98.91107940673828,
                "Value": false
            },
            "Sunglasses": {
                "Confidence": 99.7966537475586,
                "Value": false
            },
            "Gender": {
                "Confidence": 99.56611633300781,
                "Value": "Male"
            },
            "Landmarks": [
```

```
                {
                    "Y": 0.26721030473709106,
                    "X": 0.6204193830490112,
                    "Type": "eyeLeft"
                },
                {
                    "Y": 0.26831310987472534,
                    "X": 0.6776827573776245,
                    "Type": "eyeRight"
                },
                {
                    "Y": 0.3514654338359833,
                    "X": 0.6241428852081299,
                    "Type": "mouthLeft"
                },
                {
                    "Y": 0.35258132219314575,
                    "X": 0.6713621020317078,
                    "Type": "mouthRight"
                },
                {
                    "Y": 0.3140771687030792,
                    "X": 0.6428444981575012,
                    "Type": "nose"
                },
                {
                    "Y": 0.24662546813488007,
                    "X": 0.6001564860343933,
                    "Type": "leftEyeBrowLeft"
                },
                {
                    "Y": 0.24326619505882263,
                    "X": 0.6303644776344299,
                    "Type": "leftEyeBrowRight"
                },
                {
                    "Y": 0.23818562924861908,
                    "X": 0.6146903038024902,
                    "Type": "leftEyeBrowUp"
                },
                {
                    "Y": 0.24373626708984375,
                    "X": 0.6640064716339111,
                    "Type": "rightEyeBrowLeft"
```

```
                },
                {
                    "Y": 0.24877218902111053,
                    "X": 0.7025929093360901,
                    "Type": "rightEyeBrowRight"
                },
                {
                    "Y": 0.23938551545143127,
                    "X": 0.6823262572288513,
                    "Type": "rightEyeBrowUp"
                },
                {
                    "Y": 0.265746533870697,
                    "X": 0.6112898588180542,
                    "Type": "leftEyeLeft"
                },
                {
                    "Y": 0.2676128149032593,
                    "X": 0.6317071914672852,
                    "Type": "leftEyeRight"
                },
                {
                    "Y": 0.262735515832901,
                    "X": 0.6201658248901367,
                    "Type": "leftEyeUp"
                },
                {
                    "Y": 0.27025148272514343,
                    "X": 0.6206279993057251,
                    "Type": "leftEyeDown"
                },
                {
                    "Y": 0.268223375082016,
                    "X": 0.6658390760421753,
                    "Type": "rightEyeLeft"
                },
                {
                    "Y": 0.2672517001628876,
                    "X": 0.687832236289978,
                    "Type": "rightEyeRight"
                },
                {
                    "Y": 0.26383838057518005,
                    "X": 0.6769183874130249,
```

```
                    "Type": "rightEyeUp"
                },
                {
                    "Y": 0.27138751745224,
                    "X": 0.676596462726593,
                    "Type": "rightEyeDown"
                },
                {
                    "Y": 0.32283174991607666,
                    "X": 0.6350004076957703,
                    "Type": "noseLeft"
                },
                {
                    "Y": 0.3219289481639862,
                    "X": 0.6567046642303467,
                    "Type": "noseRight"
                },
                {
                    "Y": 0.3420318365097046,
                    "X": 0.6450609564781189,
                    "Type": "mouthUp"
                },
                {
                    "Y": 0.3664324879646301,
                    "X": 0.6455618143081665,
                    "Type": "mouthDown"
                },
                {
                    "Y": 0.26721030473709106,
                    "X": 0.6204193830490112,
                    "Type": "leftPupil"
                },
                {
                    "Y": 0.26831310987472534,
                    "X": 0.6776827573776245,
                    "Type": "rightPupil"
                },
                {
                    "Y": 0.26343393325805664,
                    "X": 0.5946047306060791,
                    "Type": "upperJawlineLeft"
                },
                {
                    "Y": 0.3543180525302887,
```

```
                    "X": 0.6044883728027344,
                    "Type": "midJawlineLeft"
                },
                {
                    "Y": 0.4084877669811249,
                    "X": 0.6477024555206299,
                    "Type": "chinBottom"
                },
                {
                    "Y": 0.3562754988670349,
                    "X": 0.707981526851654,
                    "Type": "midJawlineRight"
                },
                {
                    "Y": 0.26580461859703064,
                    "X": 0.7234612107276917,
                    "Type": "upperJawlineRight"
                }
            ],
            "Pose": {
                "Yaw": -3.7351467609405518,
                "Roll": -0.10309021919965744,
                "Pitch": 0.8637830018997192
            },
            "Emotions": [
                {
                    "Confidence": 8.74203109741211,
                    "Type": "SURPRISED"
                },
                {
                    "Confidence": 2.501944065093994,
                    "Type": "ANGRY"
                },
                {
                    "Confidence": 0.7378743290901184,
                    "Type": "DISGUSTED"
                },
                {
                    "Confidence": 3.5296201705932617,
                    "Type": "HAPPY"
                },
                {
                    "Confidence": 1.7162904739379883,
                    "Type": "SAD"
```

```
                },
                {
                    "Confidence": 9.518536567687988,
                    "Type": "CONFUSED"
                },
                {
                    "Confidence": 0.45474427938461304,
                    "Type": "FEAR"
                },
                {
                    "Confidence": 72.79895782470703,
                    "Type": "CALM"
                }
            ],
            "AgeRange": {
                "High": 48,
                "Low": 32
            },
            "EyesOpen": {
                "Confidence": 98.93987274169922,
                "Value": true
            },
            "BoundingBox": {
                "Width": 0.12368916720151901,
                "Top": 0.16007372736930847,
                "Left": 0.5901257991790771,
                "Height": 0.25140416622161865
            },
            "Smile": {
                "Confidence": 93.4493179321289,
                "Value": false
            },
            "MouthOpen": {
                "Confidence": 90.53053283691406,
                "Value": false
            },
            "Quality": {
                "Sharpness": 95.51618957519531,
                "Brightness": 65.29893493652344
            },
            "Mustache": {
                "Confidence": 89.85221099853516,
                "Value": false
            },
```

```
            "Beard": {
                "Confidence": 86.1991195678711,
                "Value": true
            }
        }
    ]
}
```

Para obtener más información, consulte Detección de rostros en una imagen en la Guía para desarrolladores de Amazon Rekognition.

- Para obtener más información sobre la API, consulta DetectFaceslа Referencia de AWS CLI comandos.

Java

SDK para Java 2.x

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.*;

import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DetectFaces {
    public static void main(String[] args) {
```

```
        final String usage = """

            Usage:   <bucketName> <sourceImage>

            Where:
                bucketName = The name of the Amazon S3 bucket where the source
image is stored.
                sourceImage - The name of the source image file in the Amazon S3
bucket. (for example, pic1.png).\s
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String sourceImage = args[1];
        Region region = Region.US_WEST_2;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        detectFacesinImage(rekClient, bucketName, sourceImage);
        rekClient.close();
    }

    /**
     * Detects faces in an image stored in an Amazon S3 bucket using the Amazon
Rekognition service.
     *
     * @param rekClient    The Amazon Rekognition client used to interact with
the Rekognition service.
     * @param bucketName   The name of the Amazon S3 bucket where the source
image is stored.
     * @param sourceImage  The name of the source image file in the Amazon S3
bucket.
     */
    public static void detectFacesinImage(RekognitionClient rekClient, String
bucketName, String sourceImage) {
        try {
            S3Object s3ObjectTarget = S3Object.builder()
                .bucket(bucketName)
                .name(sourceImage)
```

```
                .build();

            Image targetImage = Image.builder()
                .s3Object(s3ObjectTarget)
                .build();

            DetectFacesRequest facesRequest = DetectFacesRequest.builder()
                .attributes(Attribute.ALL)
                .image(targetImage)
                .build();

            DetectFacesResponse facesResponse =
 rekClient.detectFaces(facesRequest);
            List<FaceDetail> faceDetails = facesResponse.faceDetails();
            for (FaceDetail face : faceDetails) {
                AgeRange ageRange = face.ageRange();
                System.out.println("The detected face is estimated to be between
 "
                        + ageRange.low().toString() + " and " +
 ageRange.high().toString()
                        + " years old.");

                System.out.println("There is a smile : " +
 face.smile().value().toString());
            }

        } catch (RekognitionException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Para obtener más información sobre la API, consulta DetectFaceslaReferencia AWS SDK for Java 2.x de la API.

Kotlin

SDK para Kotlin

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun detectFacesinImage(sourceImage: String?) {
    val souImage =
        Image {
            bytes = (File(sourceImage).readBytes())
        }

    val request =
        DetectFacesRequest {
            attributes = listOf(Attribute.All)
            image = souImage
        }

    RekognitionClient.fromEnvironment { region = "us-east-1" }.use { rekClient ->
        val response = rekClient.detectFaces(request)
        response.faceDetails?.forEach { face ->
            val ageRange = face.ageRange
            println("The detected face is estimated to be between
 ${ageRange?.low} and ${ageRange?.high} years old.")
            println("There is a smile ${face.smile?.value}")
        }
    }
}
```

- Para obtener más información sobre la API, consulta DetectFaceslа referencia sobre el AWS SDK para la API de Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
class RekognitionImage:
    """
    Encapsulates an Amazon Rekognition image. This class is a thin wrapper
    around parts of the Boto3 Amazon Rekognition API.
    """

    def __init__(self, image, image_name, rekognition_client):
        """
        Initializes the image object.

        :param image: Data that defines the image, either the image bytes or
                      an Amazon S3 bucket and object key.
        :param image_name: The name of the image.
        :param rekognition_client: A Boto3 Rekognition client.
        """
        self.image = image
        self.image_name = image_name
        self.rekognition_client = rekognition_client


    def detect_faces(self):
        """
        Detects faces in the image.

        :return: The list of faces found in the image.
        """
        try:
            response = self.rekognition_client.detect_faces(
                Image=self.image, Attributes=["ALL"]
            )
            faces = [RekognitionFace(face) for face in response["FaceDetails"]]
            logger.info("Detected %s faces.", len(faces))
```

```
        except ClientError:
            logger.exception("Couldn't detect faces in %s.", self.image_name)
            raise
        else:
            return faces
```

- Para obtener más información sobre la API, consulta DetectFaceslа AWS Referencia de API de SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
    TRY.
        " Create S3 object reference for the image
        DATA(lo_s3object) = NEW /aws1/cl_reks3object(
          iv_bucket = iv_s3_bucket
          iv_name = iv_s3_key ).

        " Create image object
        DATA(lo_image) = NEW /aws1/cl_rekimage(
          io_s3object = lo_s3object ).

        " Detect faces in the image with all attributes
        DATA(lt_attributes) = VALUE /aws1/cl_rekattributes_w=>tt_attributes( ).
        DATA(lo_attr_wrapper) = NEW /aws1/cl_rekattributes_w( iv_value = 'ALL' ).
        INSERT lo_attr_wrapper INTO TABLE lt_attributes.

        oo_result = lo_rek->detectfaces(
          io_image = lo_image
          it_attributes = lt_attributes ).

        DATA(lt_face_details) = oo_result->get_facedetails( ).
```

```
      DATA(lv_detected_count) = lines( lt_face_details ).
      DATA(lv_msg8) = |{ lv_detected_count } face(s) detected in image.|.
      MESSAGE lv_msg8 TYPE 'I'.
    CATCH /aws1/cx_rekinvalids3objectex.
      MESSAGE 'Invalid S3 object.' TYPE 'E'.
    CATCH /aws1/cx_rekinvalidparameterex.
      MESSAGE 'Invalid parameter value.' TYPE 'E'.
  ENDTRY.
```

- Para obtener más información sobre la API, consulte DetectFaceslа referencia sobre la API ABAP del AWS SDK para SAP.

Para obtener una lista completa de guías para desarrolladores del AWS SDK y ejemplos de código, consulteUso de Rekognition con un SDK AWS. En este tema también se incluye información sobre cómo comenzar a utilizar el SDK y detalles sobre sus versiones anteriores.

Úselo **`DetectLabels`** con un AWS SDK o CLI

Los siguientes ejemplos de código muestran cómo utilizar `DetectLabels`.

Para obtener más información, consulte Detección de etiquetas en una imagen.

.NET

SDK para .NET

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
using System;
using System.Threading.Tasks;
using Amazon.Rekognition;
using Amazon.Rekognition.Model;

/// <summary>
/// Uses the Amazon Rekognition Service to detect labels within an image
/// stored in an Amazon Simple Storage Service (Amazon S3) bucket.
```

```
    /// </summary>
    public class DetectLabels
    {
        public static async Task Main()
        {
            string photo = "del_river_02092020_01.jpg"; // "input.jpg";
            string bucket = "amzn-s3-demo-bucket"; // "bucket";

            var rekognitionClient = new AmazonRekognitionClient();

            var detectlabelsRequest = new DetectLabelsRequest
            {
                Image = new Image()
                {
                    S3Object = new S3Object()
                    {
                        Name = photo,
                        Bucket = bucket,
                    },
                },
                MaxLabels = 10,
                MinConfidence = 75F,
            };

            try
            {
                DetectLabelsResponse detectLabelsResponse = await
rekognitionClient.DetectLabelsAsync(detectlabelsRequest);
                Console.WriteLine("Detected labels for " + photo);
                foreach (Label label in detectLabelsResponse.Labels)
                {
                    Console.WriteLine($"Name: {label.Name} Confidence:
{label.Confidence}");
                }
            }
            catch (Exception ex)
            {
                Console.WriteLine(ex.Message);
            }
        }
    }
```

Detecte las etiquetas en un archivo de imagen que está almacenado en el ordenador.

```
   using System;
   using System.IO;
   using System.Threading.Tasks;
   using Amazon.Rekognition;
   using Amazon.Rekognition.Model;

   /// <summary>
   /// Uses the Amazon Rekognition Service to detect labels within an image
   /// stored locally.
   /// </summary>
   public class DetectLabelsLocalFile
   {
       public static async Task Main()
       {
           string photo = "input.jpg";

           var image = new Amazon.Rekognition.Model.Image();
           try
           {
               using var fs = new FileStream(photo, FileMode.Open,
FileAccess.Read);
               byte[] data = null;
               data = new byte[fs.Length];
               fs.Read(data, 0, (int)fs.Length);
               image.Bytes = new MemoryStream(data);
           }
           catch (Exception)
           {
               Console.WriteLine("Failed to load file " + photo);
               return;
           }

           var rekognitionClient = new AmazonRekognitionClient();

           var detectlabelsRequest = new DetectLabelsRequest
           {
               Image = image,
               MaxLabels = 10,
               MinConfidence = 77F,
           };

           try
```

```
            {
                DetectLabelsResponse detectLabelsResponse = await
 rekognitionClient.DetectLabelsAsync(detectlabelsRequest);
                Console.WriteLine($"Detected labels for {photo}");
                foreach (Label label in detectLabelsResponse.Labels)
                {
                    Console.WriteLine($"{label.Name}: {label.Confidence}");
                }
            }
            catch (Exception ex)
            {
                Console.WriteLine(ex.Message);
            }
        }
    }
```

- Para obtener más información sobre la API, consulta DetectLabelsla Referencia AWS SDK para .NET de la API.

C++

SDK para C++

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
//! Detect instances of real-world entities within an image by using Amazon
 Rekognition
/*!
  \param imageBucket: The Amazon Simple Storage Service (Amazon S3) bucket
 containing an image.
  \param imageKey: The Amazon S3 key of an image object.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::Rekognition::detectLabels(const Aws::String &imageBucket,
```

```
                                        const Aws::String &imageKey,
                                        const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::Rekognition::RekognitionClient rekognitionClient(clientConfiguration);

    Aws::Rekognition::Model::DetectLabelsRequest request;
    Aws::Rekognition::Model::S3Object s3Object;
    s3Object.SetBucket(imageBucket);
    s3Object.SetName(imageKey);

    Aws::Rekognition::Model::Image image;
    image.SetS3Object(s3Object);

    request.SetImage(image);

    const Aws::Rekognition::Model::DetectLabelsOutcome outcome =
 rekognitionClient.DetectLabels(request);

    if (outcome.IsSuccess()) {
        const Aws::Vector<Aws::Rekognition::Model::Label> &labels =
 outcome.GetResult().GetLabels();
        if (labels.empty()) {
            std::cout << "No labels detected" << std::endl;
        } else {
            for (const Aws::Rekognition::Model::Label &label: labels) {
                std::cout << label.GetName() << ": " << label.GetConfidence() <<
 std::endl;
            }
        }
    } else {
        std::cerr << "Error while detecting labels: '"
                  << outcome.GetError().GetMessage()
                  << "'" << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Para obtener más información sobre la API, consulta DetectLabelsla Referencia AWS SDK para C++ de la API.

CLI

AWS CLI

Detección de una etiqueta en una imagen

En el siguiente ejemplo de `detect-labels` se detectan escenas y objetos en una imagen almacenada en un bucket de Amazon S3.

```
aws rekognition detect-labels \
    --image '{"S3Object":{"Bucket":"bucket","Name":"image"}}'
```

Salida:

```
{
    "Labels": [
        {
            "Instances": [],
            "Confidence": 99.15271759033203,
            "Parents": [
                {
                    "Name": "Vehicle"
                },
                {
                    "Name": "Transportation"
                }
            ],
            "Name": "Automobile"
        },
        {
            "Instances": [],
            "Confidence": 99.15271759033203,
            "Parents": [
                {
                    "Name": "Transportation"
                }
            ],
            "Name": "Vehicle"
        },
        {
            "Instances": [],
            "Confidence": 99.15271759033203,
            "Parents": [],
```

```
            "Name": "Transportation"
        },
        {
            "Instances": [
                {
                    "BoundingBox": {
                        "Width": 0.10616336017847061,
                        "Top": 0.5039216876029968,
                        "Left": 0.0037978808395564556,
                        "Height": 0.18528179824352264
                    },
                    "Confidence": 99.15271759033203
                },
                {
                    "BoundingBox": {
                        "Width": 0.2429988533258438,
                        "Top": 0.5251884460449219,
                        "Left": 0.7309805154800415,
                        "Height": 0.21577216684818268
                    },
                    "Confidence": 99.1286392211914
                },
                {
                    "BoundingBox": {
                        "Width": 0.14233611524105072,
                        "Top": 0.5333095788955688,
                        "Left": 0.6494812965393066,
                        "Height": 0.15528248250484467
                    },
                    "Confidence": 98.48368072509766
                },
                {
                    "BoundingBox": {
                        "Width": 0.11086395382881165,
                        "Top": 0.5354844927787781,
                        "Left": 0.10355594009160995,
                        "Height": 0.10271988064050674
                    },
                    "Confidence": 96.45606231689453
                },
                {
                    "BoundingBox": {
                        "Width": 0.06254628300666809,
                        "Top": 0.5573825240135193,
```

```
                "Left": 0.46083059906959534,
                "Height": 0.053911514580249786
            },
            "Confidence": 93.65448760986328
        },
        {
            "BoundingBox": {
                "Width": 0.10105438530445099,
                "Top": 0.534368634223938,
                "Left": 0.5743985772132874,
                "Height": 0.12226245552301407
            },
            "Confidence": 93.06217193603516
        },
        {
            "BoundingBox": {
                "Width": 0.056389667093753815,
                "Top": 0.5235804319381714,
                "Left": 0.9427769780158997,
                "Height": 0.17163699865341187
            },
            "Confidence": 92.6864013671875
        },
        {
            "BoundingBox": {
                "Width": 0.06003860384225845,
                "Top": 0.5441341400146484,
                "Left": 0.22409997880458832,
                "Height": 0.06737709045410156
            },
            "Confidence": 90.4227066040039
        },
        {
            "BoundingBox": {
                "Width": 0.02848697081208229,
                "Top": 0.5107086896896362,
                "Left": 0,
                "Height": 0.19150497019290924
            },
            "Confidence": 86.65286254882812
        },
        {
            "BoundingBox": {
                "Width": 0.04067881405353546,
```

```
                        "Top": 0.5566273927688599,
                        "Left": 0.316415935754776,
                        "Height": 0.03428703173995018
                    },
                    "Confidence": 85.36471557617188
                },
                {
                    "BoundingBox": {
                        "Width": 0.043411049991846085,
                        "Top": 0.5394920110702515,
                        "Left": 0.18293385207653046,
                        "Height": 0.0893595889210701
                    },
                    "Confidence": 82.21705627441406
                },
                {
                    "BoundingBox": {
                        "Width": 0.031183116137981415,
                        "Top": 0.5579366683959961,
                        "Left": 0.2853088080883026,
                        "Height": 0.03989990055561066
                    },
                    "Confidence": 81.0157470703125
                },
                {
                    "BoundingBox": {
                        "Width": 0.031113790348172188,
                        "Top": 0.5504819750785828,
                        "Left": 0.2580395042896271,
                        "Height": 0.056484755128622055
                    },
                    "Confidence": 56.13441467285156
                },
                {
                    "BoundingBox": {
                        "Width": 0.08586374670267105,
                        "Top": 0.5438792705535889,
                        "Left": 0.5128012895584106,
                        "Height": 0.08550430089235306
                    },
                    "Confidence": 52.37760925292969
                }
            ],
            "Confidence": 99.15271759033203,
```

```
            "Parents": [
                {
                    "Name": "Vehicle"
                },
                {
                    "Name": "Transportation"
                }
            ],
            "Name": "Car"
        },
        {
            "Instances": [],
            "Confidence": 98.9914321899414,
            "Parents": [],
            "Name": "Human"
        },
        {
            "Instances": [
                {
                    "BoundingBox": {
                        "Width": 0.19360728561878204,
                        "Top": 0.35072067379951477,
                        "Left": 0.43734854459762573,
                        "Height": 0.2742200493812561
                    },
                    "Confidence": 98.9914321899414
                },
                {
                    "BoundingBox": {
                        "Width": 0.03801717236638069,
                        "Top": 0.5010883808135986,
                        "Left": 0.9155802130699158,
                        "Height": 0.06597328186035156
                    },
                    "Confidence": 85.02790832519531
                }
            ],
            "Confidence": 98.9914321899414,
            "Parents": [],
            "Name": "Person"
        },
        {
            "Instances": [],
            "Confidence": 93.24951934814453,
```

```
            "Parents": [],
            "Name": "Machine"
        },
        {
            "Instances": [
                {
                    "BoundingBox": {
                        "Width": 0.03561960905790329,
                        "Top": 0.6468243598937988,
                        "Left": 0.7850857377052307,
                        "Height": 0.08878646790981293
                    },
                    "Confidence": 93.24951934814453
                },
                {
                    "BoundingBox": {
                        "Width": 0.02217046171426773,
                        "Top": 0.6149078607559204,
                        "Left": 0.04757237061858177,
                        "Height": 0.07136218994855881
                    },
                    "Confidence": 91.5025863647461
                },
                {
                    "BoundingBox": {
                        "Width": 0.016197510063648224,
                        "Top": 0.6274210214614868,
                        "Left": 0.6472989320755005,
                        "Height": 0.04955997318029404
                    },
                    "Confidence": 85.14686584472656
                },
                {
                    "BoundingBox": {
                        "Width": 0.020207518711686134,
                        "Top": 0.6348286867141724,
                        "Left": 0.7295016646385193,
                        "Height": 0.07059963047504425
                    },
                    "Confidence": 83.34547424316406
                },
                {
                    "BoundingBox": {
                        "Width": 0.020280985161662102,
```

```
                "Top": 0.6171894669532776,
                "Left": 0.08744934946298599,
                "Height": 0.05297485366463661
            },
            "Confidence": 79.9981460571289
        },
        {
            "BoundingBox": {
                "Width": 0.018318990245461464,
                "Top": 0.623889148235321,
                "Left": 0.6836880445480347,
                "Height": 0.06730121374130249
            },
            "Confidence": 78.87144470214844
        },
        {
            "BoundingBox": {
                "Width": 0.021310249343514442,
                "Top": 0.6167286038398743,
                "Left": 0.004064912907779217,
                "Height": 0.08317798376083374
            },
            "Confidence": 75.89361572265625
        },
        {
            "BoundingBox": {
                "Width": 0.03604431077837944,
                "Top": 0.7030032277107239,
                "Left": 0.9254803657531738,
                "Height": 0.04569442570209503
            },
            "Confidence": 64.402587890625
        },
        {
            "BoundingBox": {
                "Width": 0.009834849275648594,
                "Top": 0.5821820497512817,
                "Left": 0.28094568848609924,
                "Height": 0.01964157074689865
            },
            "Confidence": 62.79907989501953
        },
        {
            "BoundingBox": {
```

```
                    "Width": 0.01475677452981472,
                    "Top": 0.6137543320655823,
                    "Left": 0.5950819253921509,
                    "Height": 0.039063986390829086
                },
                "Confidence": 59.40483474731445
            }
        ],
        "Confidence": 93.24951934814453,
        "Parents": [
            {
                "Name": "Machine"
            }
        ],
        "Name": "Wheel"
    },
    {
        "Instances": [],
        "Confidence": 92.61514282226562,
        "Parents": [],
        "Name": "Road"
    },
    {
        "Instances": [],
        "Confidence": 92.37877655029297,
        "Parents": [
            {
                "Name": "Person"
            }
        ],
        "Name": "Sport"
    },
    {
        "Instances": [],
        "Confidence": 92.37877655029297,
        "Parents": [
            {
                "Name": "Person"
            }
        ],
        "Name": "Sports"
    },
    {
        "Instances": [
```

```
                {
                    "BoundingBox": {
                        "Width": 0.12326609343290329,
                        "Top": 0.6332163214683533,
                        "Left": 0.44815489649772644,
                        "Height": 0.058117982000112534
                    },
                    "Confidence": 92.37877655029297
                }
            ],
            "Confidence": 92.37877655029297,
            "Parents": [
                {
                    "Name": "Person"
                },
                {
                    "Name": "Sport"
                }
            ],
            "Name": "Skateboard"
        },
        {
            "Instances": [],
            "Confidence": 90.62931060791016,
            "Parents": [
                {
                    "Name": "Person"
                }
            ],
            "Name": "Pedestrian"
        },
        {
            "Instances": [],
            "Confidence": 88.81334686279297,
            "Parents": [],
            "Name": "Asphalt"
        },
        {
            "Instances": [],
            "Confidence": 88.81334686279297,
            "Parents": [],
            "Name": "Tarmac"
        },
        {
```

```
            "Instances": [],
            "Confidence": 88.23201751708984,
            "Parents": [],
            "Name": "Path"
        },
        {
            "Instances": [],
            "Confidence": 80.26520538330078,
            "Parents": [],
            "Name": "Urban"
        },
        {
            "Instances": [],
            "Confidence": 80.26520538330078,
            "Parents": [
                {
                    "Name": "Building"
                },
                {
                    "Name": "Urban"
                }
            ],
            "Name": "Town"
        },
        {
            "Instances": [],
            "Confidence": 80.26520538330078,
            "Parents": [],
            "Name": "Building"
        },
        {
            "Instances": [],
            "Confidence": 80.26520538330078,
            "Parents": [
                {
                    "Name": "Building"
                },
                {
                    "Name": "Urban"
                }
            ],
            "Name": "City"
        },
        {
```

```
            "Instances": [],
            "Confidence": 78.37934875488281,
            "Parents": [
                {
                    "Name": "Car"
                },
                {
                    "Name": "Vehicle"
                },
                {
                    "Name": "Transportation"
                }
            ],
            "Name": "Parking Lot"
        },
        {
            "Instances": [],
            "Confidence": 78.37934875488281,
            "Parents": [
                {
                    "Name": "Car"
                },
                {
                    "Name": "Vehicle"
                },
                {
                    "Name": "Transportation"
                }
            ],
            "Name": "Parking"
        },
        {
            "Instances": [],
            "Confidence": 74.37590026855469,
            "Parents": [
                {
                    "Name": "Building"
                },
                {
                    "Name": "Urban"
                },
                {
                    "Name": "City"
                }
```

```
            ],
            "Name": "Downtown"
        },
        {
            "Instances": [],
            "Confidence": 69.84622955322266,
            "Parents": [
                {
                    "Name": "Road"
                }
            ],
            "Name": "Intersection"
        },
        {
            "Instances": [],
            "Confidence": 57.68518829345703,
            "Parents": [
                {
                    "Name": "Sports Car"
                },
                {
                    "Name": "Car"
                },
                {
                    "Name": "Vehicle"
                },
                {
                    "Name": "Transportation"
                }
            ],
            "Name": "Coupe"
        },
        {
            "Instances": [],
            "Confidence": 57.68518829345703,
            "Parents": [
                {
                    "Name": "Car"
                },
                {
                    "Name": "Vehicle"
                },
                {
                    "Name": "Transportation"
```

```
                }
            ],
            "Name": "Sports Car"
        },
        {
            "Instances": [],
            "Confidence": 56.59492111206055,
            "Parents": [
                {
                    "Name": "Path"
                }
            ],
            "Name": "Sidewalk"
        },
        {
            "Instances": [],
            "Confidence": 56.59492111206055,
            "Parents": [
                {
                    "Name": "Path"
                }
            ],
            "Name": "Pavement"
        },
        {
            "Instances": [],
            "Confidence": 55.58770751953125,
            "Parents": [
                {
                    "Name": "Building"
                },
                {
                    "Name": "Urban"
                }
            ],
            "Name": "Neighborhood"
        }
    ],
    "LabelModelVersion": "2.0"
}
```


Para obtener más información, consulte Detección de etiquetas en una imagen en la Guía para desarrolladores de Amazon Rekognition.

- Para obtener más información sobre la API, consulta DetectLabelsla Referencia de AWS CLI comandos.

Java

SDK para Java 2.x

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.*;

import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DetectLabels {
    public static void main(String[] args) {
        final String usage = """
            Usage: <bucketName> <sourceImage>

            Where:
                bucketName - The name of the Amazon S3 bucket where the image is
 stored
```

```
                sourceImage - The name of the image file (for example, pic1.png).
\s
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0] ;
        String sourceImage = args[1] ;
        Region region = Region.US_WEST_2;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        detectImageLabels(rekClient, bucketName, sourceImage);
        rekClient.close();
    }

    /**
     * Detects the labels in an image stored in an Amazon S3 bucket using the
 Amazon Rekognition service.
     *
     * @param rekClient     the Amazon Rekognition client used to make the
 detection request
     * @param bucketName    the name of the Amazon S3 bucket where the image is
 stored
     * @param sourceImage   the name of the image file to be analyzed
     */
    public static void detectImageLabels(RekognitionClient rekClient, String
 bucketName, String sourceImage) {
        try {
            S3Object s3ObjectTarget = S3Object.builder()
                    .bucket(bucketName)
                    .name(sourceImage)
                    .build();

            Image souImage = Image.builder()
                    .s3Object(s3ObjectTarget)
                    .build();

            DetectLabelsRequest detectLabelsRequest =
 DetectLabelsRequest.builder()
```

```
                    .image(souImage)
                    .maxLabels(10)
                    .build();

            DetectLabelsResponse labelsResponse =
 rekClient.detectLabels(detectLabelsRequest);
            List<Label> labels = labelsResponse.labels();
            System.out.println("Detected labels for the given photo");
            for (Label label : labels) {
                System.out.println(label.name() + ": " +
 label.confidence().toString());
            }

        } catch (RekognitionException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Para obtener más información sobre la API, consulta DetectLabelsla Referencia AWS SDK for Java 2.x de la API.

Kotlin

SDK para Kotlin

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun detectImageLabels(sourceImage: String) {
    val souImage =
        Image {
            bytes = (File(sourceImage).readBytes())
        }
    val request =
        DetectLabelsRequest {
```

```
            image = souImage
            maxLabels = 10
        }

    RekognitionClient.fromEnvironment { region = "us-east-1" }.use { rekClient ->
        val response = rekClient.detectLabels(request)
        response.labels?.forEach { label ->
            println("${label.name} : ${label.confidence}")
        }
    }
}
```

- Para obtener más información sobre la API, consulta DetectLabelsla referencia sobre el AWS SDK para la API de Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
class RekognitionImage:
    """
    Encapsulates an Amazon Rekognition image. This class is a thin wrapper
    around parts of the Boto3 Amazon Rekognition API.
    """

    def __init__(self, image, image_name, rekognition_client):
        """
        Initializes the image object.

        :param image: Data that defines the image, either the image bytes or
                      an Amazon S3 bucket and object key.
        :param image_name: The name of the image.
        :param rekognition_client: A Boto3 Rekognition client.
        """
```

```
        self.image = image
        self.image_name = image_name
        self.rekognition_client = rekognition_client


    def detect_labels(self, max_labels):
        """
        Detects labels in the image. Labels are objects and people.

        :param max_labels: The maximum number of labels to return.
        :return: The list of labels detected in the image.
        """
        try:
            response = self.rekognition_client.detect_labels(
                Image=self.image, MaxLabels=max_labels
            )
            labels = [RekognitionLabel(label) for label in response["Labels"]]
            logger.info("Found %s labels in %s.", len(labels), self.image_name)
        except ClientError:
            logger.info("Couldn't detect labels in %s.", self.image_name)
            raise
        else:
            return labels
```

- Para obtener más información sobre la API, consulta DetectLabelsla AWS Referencia de API de SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
    TRY.
        " Create S3 object reference for the image
```

```
        DATA(lo_s3object) = NEW /aws1/cl_reks3object(
          iv_bucket = iv_s3_bucket
          iv_name = iv_s3_key ).

        " Create image object
        DATA(lo_image) = NEW /aws1/cl_rekimage(
          io_s3object = lo_s3object ).

        " Detect labels in the image
        oo_result = lo_rek->detectlabels(
          io_image = lo_image
          iv_maxlabels = iv_max_labels ).

        DATA(lt_labels) = oo_result->get_labels( ).
        DATA(lv_label_count) = lines( lt_labels ).
        DATA(lv_msg9) = |{ lv_label_count } label(s) detected in image.|.
        MESSAGE lv_msg9 TYPE 'I'.
      CATCH /aws1/cx_rekinvalids3objectex.
        MESSAGE 'Invalid S3 object.' TYPE 'E'.
      CATCH /aws1/cx_rekinvalidparameterex.
        MESSAGE 'Invalid parameter value.' TYPE 'E'.
    ENDTRY.
```

- Para obtener más información sobre la API, consulte DetectLabelsla referencia sobre la API ABAP del AWS SDK para SAP.

Para obtener una lista completa de guías para desarrolladores del AWS SDK y ejemplos de código, consulteUso de Rekognition con un SDK AWS. En este tema también se incluye información sobre cómo comenzar a utilizar el SDK y detalles sobre sus versiones anteriores.

Úselo **`DetectModerationLabels`** con un AWS SDK o CLI

Los siguientes ejemplos de código muestran cómo utilizar `DetectModerationLabels`.

Para obtener más información, consulte Detección de imágenes inapropiadas.

.NET

SDK para .NET

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
    using System;
    using System.Threading.Tasks;
    using Amazon.Rekognition;
    using Amazon.Rekognition.Model;

    /// <summary>
    /// Uses the Amazon Rekognition Service to detect unsafe content in a
    /// JPEG or PNG format image.
    /// </summary>
    public class DetectModerationLabels
    {
        public static async Task Main(string[] args)
        {
            string photo = "input.jpg";
            string bucket = "amzn-s3-demo-bucket";

            var rekognitionClient = new AmazonRekognitionClient();

            var detectModerationLabelsRequest = new
DetectModerationLabelsRequest()
            {
                Image = new Image()
                {
                    S3Object = new S3Object()
                    {
                        Name = photo,
                        Bucket = bucket,
                    },
                },
                MinConfidence = 60F,
            };
```

```
            try
            {
                var detectModerationLabelsResponse = await
rekognitionClient.DetectModerationLabelsAsync(detectModerationLabelsRequest);
                Console.WriteLine("Detected labels for " + photo);
                foreach (ModerationLabel label in
detectModerationLabelsResponse.ModerationLabels)
                {
                    Console.WriteLine($"Label: {label.Name}");
                    Console.WriteLine($"Confidence: {label.Confidence}");
                    Console.WriteLine($"Parent: {label.ParentName}");
                }
            }
            catch (Exception ex)
            {
                Console.WriteLine(ex.Message);
            }
        }
    }
```

- Para obtener más información sobre la API, consulta DetectModerationLabelsla Referencia AWS SDK para .NET de la API.

CLI

AWS CLI

Detección de contenido no seguro en una imagen

El siguiente comando `detect-moderation-labels` detecta contenido no seguro en la imagen especificada almacenada en un bucket de Amazon S3.

```
aws rekognition detect-moderation-labels \
    --image "S3Object={Bucket=MyImageS3Bucket,Name=gun.jpg}"
```

Salida:

```
{
    "ModerationModelVersion": "3.0",
    "ModerationLabels": [
```

```
        {
            "Confidence": 97.29618072509766,
            "ParentName": "Violence",
            "Name": "Weapon Violence"
        },
        {
            "Confidence": 97.29618072509766,
            "ParentName": "",
            "Name": "Violence"
        }
    ]
}
```

Para obtener más información, consulte Detección de imágenes no seguras en la Guía para desarrolladores de Amazon Rekognition.

- Para obtener más información sobre la API, consulta DetectModerationLabelsla Referencia de AWS CLI comandos.

Java

SDK para Java 2.x

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.*;

import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
```

```
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DetectModerationLabels {

    public static void main(String[] args) {
        final String usage = """
            Usage:  <bucketName>  <sourceImage>

            Where:
                bucketName - The name of the S3 bucket where the images are
 stored.
                sourceImage - The name of the image (for example, pic1.png).\s
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String sourceImage = args[1];
        Region region = Region.US_WEST_2;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        detectModLabels(rekClient, bucketName, sourceImage);
        rekClient.close();
    }

    /**
     * Detects moderation labels in an image stored in an Amazon S3 bucket.
     *
     * @param rekClient      the Amazon Rekognition client to use for the
 detection
     * @param bucketName     the name of the Amazon S3 bucket where the image is
 stored
     * @param sourceImage    the name of the image file to be analyzed
     *
     * @throws RekognitionException if there is an error during the image
 detection process
```

```
     */
    public static void detectModLabels(RekognitionClient rekClient, String
 bucketName, String sourceImage) {
        try {
            S3Object s3ObjectTarget = S3Object.builder()
                    .bucket(bucketName)
                    .name(sourceImage)
                    .build();

            Image targetImage = Image.builder()
                    .s3Object(s3ObjectTarget)
                    .build();

            DetectModerationLabelsRequest moderationLabelsRequest =
 DetectModerationLabelsRequest.builder()
                    .image(targetImage)
                    .minConfidence(60F)
                    .build();

            DetectModerationLabelsResponse moderationLabelsResponse = rekClient
                    .detectModerationLabels(moderationLabelsRequest);
            List<ModerationLabel> labels =
 moderationLabelsResponse.moderationLabels();
            System.out.println("Detected labels for image");
            for (ModerationLabel label : labels) {
                System.out.println("Label: " + label.name()
                        + "\n Confidence: " + label.confidence().toString() + "%"
                        + "\n Parent:" + label.parentName());
            }

        } catch (RekognitionException e) {
            e.printStackTrace();
            System.exit(1);
        }
    }
}
```

- Para obtener más información sobre la API, consulta DetectModerationLabelsla Referencia AWS SDK for Java 2.x de la API.

Kotlin

SDK para Kotlin

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun detectModLabels(sourceImage: String) {
    val myImage =
        Image {
            this.bytes = (File(sourceImage).readBytes())
        }

    val request =
        DetectModerationLabelsRequest {
            image = myImage
            minConfidence = 60f
        }

    RekognitionClient.fromEnvironment { region = "us-east-1" }.use { rekClient ->
        val response = rekClient.detectModerationLabels(request)
        response.moderationLabels?.forEach { label ->
            println("Label: ${label.name} - Confidence: ${label.confidence} %
 Parent: ${label.parentName}")
        }
    }
}
```

- Para obtener más información sobre la API, consulta DetectModerationLabelsla referencia sobre el AWS SDK para la API de Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
class RekognitionImage:
    """
    Encapsulates an Amazon Rekognition image. This class is a thin wrapper
    around parts of the Boto3 Amazon Rekognition API.
    """

    def __init__(self, image, image_name, rekognition_client):
        """
        Initializes the image object.

        :param image: Data that defines the image, either the image bytes or
                      an Amazon S3 bucket and object key.
        :param image_name: The name of the image.
        :param rekognition_client: A Boto3 Rekognition client.
        """
        self.image = image
        self.image_name = image_name
        self.rekognition_client = rekognition_client


    def detect_moderation_labels(self):
        """
        Detects moderation labels in the image. Moderation labels identify
 content
        that may be inappropriate for some audiences.

        :return: The list of moderation labels found in the image.
        """
        try:
            response = self.rekognition_client.detect_moderation_labels(
                Image=self.image
            )
```

```
            labels = [
                RekognitionModerationLabel(label)
                for label in response["ModerationLabels"]
            ]
            logger.info(
                "Found %s moderation labels in %s.", len(labels), self.image_name
            )
        except ClientError:
            logger.exception(
                "Couldn't detect moderation labels in %s.", self.image_name
            )
            raise
        else:
            return labels
```

- Para obtener más información sobre la API, consulta DetectModerationLabelsla AWS Referencia de API de SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
    TRY.
        " Create S3 object reference for the image
        DATA(lo_s3object) = NEW /aws1/cl_reks3object(
          iv_bucket = iv_s3_bucket
          iv_name = iv_s3_key ).

        " Create image object
        DATA(lo_image) = NEW /aws1/cl_rekimage(
          io_s3object = lo_s3object ).

        " Detect moderation labels
```

```
        oo_result = lo_rek->detectmoderationlabels(
          io_image = lo_image ).

        DATA(lt_moderation_labels) = oo_result->get_moderationlabels( ).
        DATA(lv_mod_count) = lines( lt_moderation_labels ).
        DATA(lv_msg10) = |{ lv_mod_count } moderation label(s) detected.|.
        MESSAGE lv_msg10 TYPE 'I'.
      CATCH /aws1/cx_rekinvalids3objectex.
        MESSAGE 'Invalid S3 object.' TYPE 'E'.
      CATCH /aws1/cx_rekinvalidparameterex.
        MESSAGE 'Invalid parameter value.' TYPE 'E'.
    ENDTRY.
```

- Para obtener más información sobre la API, consulte DetectModerationLabelsla referencia sobre la API ABAP del AWS SDK para SAP.

Para obtener una lista completa de guías para desarrolladores del AWS SDK y ejemplos de código, consulteUso de Rekognition con un SDK AWS. En este tema también se incluye información sobre cómo comenzar a utilizar el SDK y detalles sobre sus versiones anteriores.

Úselo **`DetectText`** con un AWS SDK o CLI

Los siguientes ejemplos de código muestran cómo utilizar `DetectText`.

Para obtener más información, consulte Detección de texto en una imagen.

.NET

SDK para .NET

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
using System;
using System.Threading.Tasks;
using Amazon.Rekognition;
using Amazon.Rekognition.Model;
```

```
    /// <summary>
    /// Uses the Amazon Rekognition Service to detect text in an image. The
    /// example was created using the AWS SDK for .NET version 3.7 and .NET
    /// Core 5.0.
    /// </summary>
    public class DetectText
    {
        public static async Task Main()
        {
            string photo = "Dad_photographer.jpg"; // "input.jpg";
            string bucket = "amzn-s3-demo-bucket"; // "bucket";

            var rekognitionClient = new AmazonRekognitionClient();

            var detectTextRequest = new DetectTextRequest()
            {
                Image = new Image()
                {
                    S3Object = new S3Object()
                    {
                        Name = photo,
                        Bucket = bucket,
                    },
                },
            };

            try
            {
                DetectTextResponse detectTextResponse = await
rekognitionClient.DetectTextAsync(detectTextRequest);
                Console.WriteLine($"Detected lines and words for {photo}");
                detectTextResponse.TextDetections.ForEach(text =>
                {
                    Console.WriteLine($"Detected: {text.DetectedText}");
                    Console.WriteLine($"Confidence: {text.Confidence}");
                    Console.WriteLine($"Id : {text.Id}");
                    Console.WriteLine($"Parent Id: {text.ParentId}");
                    Console.WriteLine($"Type: {text.Type}");
                });
            }
            catch (Exception e)
            {
                Console.WriteLine(e.Message);
```

```
            }
        }
    }
```

- Para obtener más información sobre la API, consulta DetectTextla Referencia AWS SDK para .NET de la API.

CLI

AWS CLI

Detección de texto en una imagen

El siguiente comando `detect-text` detecta el texto de la imagen especificada.

```
aws rekognition detect-text \
    --image '{"S3Object":
{"Bucket":"MyImageS3Bucket","Name":"ExamplePicture.jpg"}}'
```

Salida:

```
{
    "TextDetections": [
        {
            "Geometry": {
                "BoundingBox": {
                    "Width": 0.24624845385551453,
                    "Top": 0.28288066387176514,
                    "Left": 0.391388863325119,
                    "Height": 0.022687450051307678
                },
                "Polygon": [
                    {
                        "Y": 0.28288066387176514,
                        "X": 0.391388863325119
                    },
                    {
                        "Y": 0.2826388478279114,
                        "X": 0.6376373171806335
                    },
```

```
                    {
                        "Y": 0.30532628297805786,
                        "X": 0.637677013874054
                    },
                    {
                        "Y": 0.305568128824234,
                        "X": 0.39142853021621704
                    }
                ]
            },
            "Confidence": 94.35709381103516,
            "DetectedText": "ESTD 1882",
            "Type": "LINE",
            "Id": 0
        },
        {
            "Geometry": {
                "BoundingBox": {
                    "Width": 0.33933889865875244,
                    "Top": 0.32603850960731506,
                    "Left": 0.34534579515457153,
                    "Height": 0.07126858830451965
                },
                "Polygon": [
                    {
                        "Y": 0.32603850960731506,
                        "X": 0.34534579515457153
                    },
                    {
                        "Y": 0.32633158564567566,
                        "X": 0.684684693813324
                    },
                    {
                        "Y": 0.3976001739501953,
                        "X": 0.684575080871582
                    },
                    {
                        "Y": 0.3973070979118347,
                        "X": 0.345236212015152
                    }
                ]
            },
            "Confidence": 99.95779418945312,
            "DetectedText": "BRAINS",
```

```
            "Type": "LINE",
            "Id": 1
        },
        {
            "Confidence": 97.22098541259766,
            "Geometry": {
                "BoundingBox": {
                    "Width": 0.061079490929841995,
                    "Top": 0.2843210697174072,
                    "Left": 0.391391396522522,
                    "Height": 0.021029088646173477
                },
                "Polygon": [
                    {
                        "Y": 0.2843210697174072,
                        "X": 0.391391396522522
                    },
                    {
                        "Y": 0.2828207015991211,
                        "X": 0.4524524509906769
                    },
                    {
                        "Y": 0.3038259446620941,
                        "X": 0.4534534513950348
                    },
                    {
                        "Y": 0.30532634258270264,
                        "X": 0.3923923969268799
                    }
                ]
            },
            "DetectedText": "ESTD",
            "ParentId": 0,
            "Type": "WORD",
            "Id": 2
        },
        {
            "Confidence": 91.49320983886719,
            "Geometry": {
                "BoundingBox": {
                    "Width": 0.07007007300853729,
                    "Top": 0.2828207015991211,
                    "Left": 0.5675675868988037,
                    "Height": 0.02250562608242035
```

```
                },
                "Polygon": [
                    {
                        "Y": 0.2828207015991211,
                        "X": 0.5675675868988037
                    },
                    {
                        "Y": 0.2828207015991211,
                        "X": 0.6376376152038574
                    },
                    {
                        "Y": 0.30532634258270264,
                        "X": 0.6376376152038574
                    },
                    {
                        "Y": 0.30532634258270264,
                        "X": 0.5675675868988037
                    }
                ]
            },
            "DetectedText": "1882",
            "ParentId": 0,
            "Type": "WORD",
            "Id": 3
        },
        {
            "Confidence": 99.95779418945312,
            "Geometry": {
                "BoundingBox": {
                    "Width": 0.33933934569358826,
                    "Top": 0.32633158564567566,
                    "Left": 0.3453453481197357,
                    "Height": 0.07127484679222107
                },
                "Polygon": [
                    {
                        "Y": 0.32633158564567566,
                        "X": 0.3453453481197357
                    },
                    {
                        "Y": 0.32633158564567566,
                        "X": 0.684684693813324
                    },
                    {
```

```
                        "Y": 0.39759939908981323,
                        "X": 0.6836836934089661
                    },
                    {
                        "Y": 0.39684921503067017,
                        "X": 0.3453453481197357
                    }
                ]
            },
            "DetectedText": "BRAINS",
            "ParentId": 1,
            "Type": "WORD",
            "Id": 4
        }
    ]
}
```

- Para obtener más información sobre la API, consulta DetectTextla Referencia de AWS CLI comandos.

Java

SDK para Java 2.x

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.*;

import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;
import java.util.List;

/**
```

```
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DetectText {
    public static void main(String[] args) {
        final String usage = "\n" +
            "Usage:   <bucketName> <sourceImage>\n" +
            "\n" +
            "Where:\n" +
            "   bucketName - The name of the S3 bucket where the image is stored
\n" +
            "   sourceImage - The path to the image that contains text (for
 example, pic1.png). \n";

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String sourceImage = args[1];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        detectTextLabels(rekClient, bucketName, sourceImage);
        rekClient.close();
    }

    /**
     * Detects text labels in an image stored in an S3 bucket using Amazon
 Rekognition.
     *
     * @param rekClient    an instance of the Amazon Rekognition client
     * @param bucketName   the name of the S3 bucket where the image is stored
     * @param sourceImage  the name of the image file in the S3 bucket
     * @throws RekognitionException if an error occurs while calling the Amazon
 Rekognition API
```

```
     */
    public static void detectTextLabels(RekognitionClient rekClient, String
 bucketName, String sourceImage) {
        try {
            S3Object s3ObjectTarget = S3Object.builder()
                    .bucket(bucketName)
                    .name(sourceImage)
                    .build();

            Image souImage = Image.builder()
                    .s3Object(s3ObjectTarget)
                    .build();

            DetectTextRequest textRequest = DetectTextRequest.builder()
                    .image(souImage)
                    .build();

            DetectTextResponse textResponse = rekClient.detectText(textRequest);
            List<TextDetection> textCollection = textResponse.textDetections();
            System.out.println("Detected lines and words");
            for (TextDetection text : textCollection) {
                System.out.println("Detected: " + text.detectedText());
                System.out.println("Confidence: " +
 text.confidence().toString());
                System.out.println("Id : " + text.id());
                System.out.println("Parent Id: " + text.parentId());
                System.out.println("Type: " + text.type());
                System.out.println();
            }

        } catch (RekognitionException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Para obtener más información sobre la API, consulta DetectTextla Referencia AWS SDK for Java 2.x de la API.

Kotlin

SDK para Kotlin

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun detectTextLabels(sourceImage: String?) {
    val souImage =
        Image {
            bytes = (File(sourceImage).readBytes())
        }

    val request =
        DetectTextRequest {
            image = souImage
        }

    RekognitionClient.fromEnvironment { region = "us-east-1" }.use { rekClient ->
        val response = rekClient.detectText(request)
        response.textDetections?.forEach { text ->
            println("Detected: ${text.detectedText}")
            println("Confidence: ${text.confidence}")
            println("Id: ${text.id}")
            println("Parent Id:  ${text.parentId}")
            println("Type: ${text.type}")
        }
    }
}
```

- Para obtener más información sobre la API, consulta DetectTextla referencia sobre el AWS SDK para la API de Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
class RekognitionImage:
    """
    Encapsulates an Amazon Rekognition image. This class is a thin wrapper
    around parts of the Boto3 Amazon Rekognition API.
    """

    def __init__(self, image, image_name, rekognition_client):
        """
        Initializes the image object.

        :param image: Data that defines the image, either the image bytes or
                      an Amazon S3 bucket and object key.
        :param image_name: The name of the image.
        :param rekognition_client: A Boto3 Rekognition client.
        """
        self.image = image
        self.image_name = image_name
        self.rekognition_client = rekognition_client


    def detect_text(self):
        """
        Detects text in the image.

        :return The list of text elements found in the image.
        """
        try:
            response = self.rekognition_client.detect_text(Image=self.image)
            texts = [RekognitionText(text) for text in
 response["TextDetections"]]
            logger.info("Found %s texts in %s.", len(texts), self.image_name)
        except ClientError:
```

```
            logger.exception("Couldn't detect text in %s.", self.image_name)
            raise
        else:
            return texts
```

- Para obtener más información sobre la API, consulta DetectTextla AWS Referencia de API de SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
    TRY.
        " Create S3 object reference for the image
        DATA(lo_s3object) = NEW /aws1/cl_reks3object(
          iv_bucket = iv_s3_bucket
          iv_name = iv_s3_key ).

        " Create image object
        DATA(lo_image) = NEW /aws1/cl_rekimage(
          io_s3object = lo_s3object ).

        " Detect text in the image
        oo_result = lo_rek->detecttext(
          io_image = lo_image ).

        DATA(lt_text_detections) = oo_result->get_textdetections( ).
        DATA(lv_text_count) = lines( lt_text_detections ).
        DATA(lv_msg11) = |{ lv_text_count } text detection(s) found.|.
        MESSAGE lv_msg11 TYPE 'I'.
      CATCH /aws1/cx_rekinvalids3objectex.
        MESSAGE 'Invalid S3 object.' TYPE 'E'.
      CATCH /aws1/cx_rekinvalidparameterex.
```

```
        MESSAGE 'Invalid parameter value.' TYPE 'E'.
    ENDTRY.
```

- Para obtener más información sobre la API, consulte DetectTextla referencia sobre la API ABAP del AWS SDK para SAP.

Para obtener una lista completa de guías para desarrolladores del AWS SDK y ejemplos de código, consulteUso de Rekognition con un SDK AWS. En este tema también se incluye información sobre cómo comenzar a utilizar el SDK y detalles sobre sus versiones anteriores.

Úselo **`GetCelebrityInfo`** con un AWS SDK o CLI

Los siguientes ejemplos de código muestran cómo utilizar `GetCelebrityInfo`.

.NET

SDK para .NET

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
using System;
using System.Threading.Tasks;
using Amazon.Rekognition;
using Amazon.Rekognition.Model;

/// <summary>
/// Shows how to use Amazon Rekognition to retrieve information about the
/// celebrity identified by the supplied celebrity Id.
/// </summary>
public class CelebrityInfo
{
    public static async Task Main()
    {
        string celebId = "nnnnnnnn";

        var rekognitionClient = new AmazonRekognitionClient();
```

```
            var celebrityInfoRequest = new GetCelebrityInfoRequest
            {
                Id = celebId,
            };

            Console.WriteLine($"Getting information for celebrity: {celebId}");

            var celebrityInfoResponse = await
 rekognitionClient.GetCelebrityInfoAsync(celebrityInfoRequest);

            // Display celebrity information.
            Console.WriteLine($"celebrity name: {celebrityInfoResponse.Name}");
            Console.WriteLine("Further information (if available):");
            celebrityInfoResponse.Urls.ForEach(url =>
            {
                Console.WriteLine(url);
            });
        }
    }
```

- Para obtener más información sobre la API, consulta GetCelebrityInfola Referencia AWS SDK para .NET de la API.

CLI

AWS CLI

Obtención de información sobre un famoso

El siguiente comando `get-celebrity-info` muestra información sobre el famoso especificado. El parámetro `id` procede de una llamada anterior a `recognize-celebrities`.

```
aws rekognition get-celebrity-info --id nnnnnnn
```

Salida:

```
{
    "Name": "Celeb A",
    "Urls": [
```

```
        "www.imdb.com/name/aaaaaaaaa"
    ]
}
```

Para obtener más información, consulte Obtención de información sobre un famoso en la Guía para desarrolladores de Amazon Rekognition.

- Para obtener más información sobre la API, consulta GetCelebrityInfola Referencia de AWS CLI comandos.

Para obtener una lista completa de guías para desarrolladores del AWS SDK y ejemplos de código, consulteUso de Rekognition con un SDK AWS. En este tema también se incluye información sobre cómo comenzar a utilizar el SDK y detalles sobre sus versiones anteriores.

Úselo **`IndexFaces`** con un AWS SDK o CLI

Los siguientes ejemplos de código muestran cómo utilizar `IndexFaces`.

Para obtener más información, consulte Adición de rostros a una colección.

.NET

SDK para .NET

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using Amazon.Rekognition;
using Amazon.Rekognition.Model;

/// <summary>
/// Uses the Amazon Rekognition Service to detect faces in an image
/// that has been uploaded to an Amazon Simple Storage Service (Amazon S3)
/// bucket and then adds the information to a collection.
/// </summary>
```

```
    public class AddFaces
    {
        public static async Task Main()
        {
            string collectionId = "MyCollection2";
            string bucket = "amzn-s3-demo-bucket";
            string photo = "input.jpg";

            var rekognitionClient = new AmazonRekognitionClient();

            var image = new Image
            {
                S3Object = new S3Object
                {
                    Bucket = bucket,
                    Name = photo,
                },
            };

            var indexFacesRequest = new IndexFacesRequest
            {
                Image = image,
                CollectionId = collectionId,
                ExternalImageId = photo,
                DetectionAttributes = new List<string>() { "ALL" },
            };

            IndexFacesResponse indexFacesResponse = await
rekognitionClient.IndexFacesAsync(indexFacesRequest);

            Console.WriteLine($"{photo} added");
            foreach (FaceRecord faceRecord in indexFacesResponse.FaceRecords)
            {
                Console.WriteLine($"Face detected: Faceid is
{faceRecord.Face.FaceId}");
            }
        }
    }
```

- Para obtener más información sobre la API, consulta IndexFaceslla Referencia AWS SDK para .NET de la API.

CLI

AWS CLI

Adición de rostros a una colección

El siguiente comando `index-faces` añade los rostros que se encuentran en una imagen a la colección especificada.

```
aws rekognition index-faces \
    --image '{"S3Object":{"Bucket":"MyVideoS3Bucket","Name":"MyPicture.jpg"}}' \
    --collection-id MyCollection \
    --max-faces 1 \
    --quality-filter "AUTO" \
    --detection-attributes "ALL" \
    --external-image-id "MyPicture.jpg"
```

Salida:

```
{
    "FaceRecords": [
        {
            "FaceDetail": {
                "Confidence": 99.993408203125,
                "Eyeglasses": {
                    "Confidence": 99.11750030517578,
                    "Value": false
                },
                "Sunglasses": {
                    "Confidence": 99.98249053955078,
                    "Value": false
                },
                "Gender": {
                    "Confidence": 99.92769622802734,
                    "Value": "Male"
                },
                "Landmarks": [
                    {
                        "Y": 0.26750367879867554,
                        "X": 0.6202793717384338,
                        "Type": "eyeLeft"
                    },
                    {
```

```
                        "Y": 0.26642778515815735,
                        "X": 0.6787431836128235,
                        "Type": "eyeRight"
                    },
                    {
                        "Y": 0.31361380219459534,
                        "X": 0.6421601176261902,
                        "Type": "nose"
                    },
                    {
                        "Y": 0.3495299220085144,
                        "X": 0.6216195225715637,
                        "Type": "mouthLeft"
                    },
                    {
                        "Y": 0.35194727778434753,
                        "X": 0.669899046421051,
                        "Type": "mouthRight"
                    },
                    {
                        "Y": 0.26844894886016846,
                        "X": 0.6210268139839172,
                        "Type": "leftPupil"
                    },
                    {
                        "Y": 0.26707562804222107,
                        "X": 0.6817160844802856,
                        "Type": "rightPupil"
                    },
                    {
                        "Y": 0.24834522604942322,
                        "X": 0.6018546223640442,
                        "Type": "leftEyeBrowLeft"
                    },
                    {
                        "Y": 0.24397172033786774,
                        "X": 0.6172008514404297,
                        "Type": "leftEyeBrowUp"
                    },
                    {
                        "Y": 0.24677404761314392,
                        "X": 0.6339119076728821,
                        "Type": "leftEyeBrowRight"
                    },
```

```
                        {
                            "Y": 0.24582654237747192,
                            "X": 0.6619398593902588,
                            "Type": "rightEyeBrowLeft"
                        },
                        {
                            "Y": 0.23973053693771362,
                            "X": 0.6804757118225098,
                            "Type": "rightEyeBrowUp"
                        },
                        {
                            "Y": 0.24441994726657867,
                            "X": 0.6978968977928162,
                            "Type": "rightEyeBrowRight"
                        },
                        {
                            "Y": 0.2695908546447754,
                            "X": 0.6085202693939209,
                            "Type": "leftEyeLeft"
                        },
                        {
                            "Y": 0.26716896891593933,
                            "X": 0.6315826177597046,
                            "Type": "leftEyeRight"
                        },
                        {
                            "Y": 0.26289820671081543,
                            "X": 0.6202316880226135,
                            "Type": "leftEyeUp"
                        },
                        {
                            "Y": 0.27123287320137024,
                            "X": 0.6205548048019409,
                            "Type": "leftEyeDown"
                        },
                        {
                            "Y": 0.2668408751487732,
                            "X": 0.6663622260093689,
                            "Type": "rightEyeLeft"
                        },
                        {
                            "Y": 0.26741549372673035,
                            "X": 0.6910083889961243,
                            "Type": "rightEyeRight"
```

```
                    },
                    {
                        "Y": 0.2614026665687561,
                        "X": 0.6785826086997986,
                        "Type": "rightEyeUp"
                    },
                    {
                        "Y": 0.27075251936912537,
                        "X": 0.6789616942405701,
                        "Type": "rightEyeDown"
                    },
                    {
                        "Y": 0.3211299479007721,
                        "X": 0.6324167847633362,
                        "Type": "noseLeft"
                    },
                    {
                        "Y": 0.32276326417922974,
                        "X": 0.6558475494384766,
                        "Type": "noseRight"
                    },
                    {
                        "Y": 0.34385165572166443,
                        "X": 0.6444970965385437,
                        "Type": "mouthUp"
                    },
                    {
                        "Y": 0.3671635091304779,
                        "X": 0.6459195017814636,
                        "Type": "mouthDown"
                    }
                ],
                "Pose": {
                    "Yaw": -9.54541015625,
                    "Roll": -0.5709401965141296,
                    "Pitch": 0.6045494675636292
                },
                "Emotions": [
                    {
                        "Confidence": 39.90074157714844,
                        "Type": "HAPPY"
                    },
                    {
                        "Confidence": 23.38753890991211,
```

```
                        "Type": "CALM"
                    },
                    {
                        "Confidence": 5.840933322906494,
                        "Type": "CONFUSED"
                    }
                ],
                "AgeRange": {
                    "High": 63,
                    "Low": 45
                },
                "EyesOpen": {
                    "Confidence": 99.80887603759766,
                    "Value": true
                },
                "BoundingBox": {
                    "Width": 0.18562500178813934,
                    "Top": 0.1618015021085739,
                    "Left": 0.5575000047683716,
                    "Height": 0.24770642817020416
                },
                "Smile": {
                    "Confidence": 99.69740295410156,
                    "Value": false
                },
                "MouthOpen": {
                    "Confidence": 99.97393798828125,
                    "Value": false
                },
                "Quality": {
                    "Sharpness": 95.54405975341797,
                    "Brightness": 63.867706298828125
                },
                "Mustache": {
                    "Confidence": 97.05007934570312,
                    "Value": false
                },
                "Beard": {
                    "Confidence": 87.34505462646484,
                    "Value": false
                }
            },
            "Face": {
                "BoundingBox": {
```

```
                    "Width": 0.18562500178813934,
                    "Top": 0.1618015021085739,
                    "Left": 0.5575000047683716,
                    "Height": 0.24770642817020416
                },
                "FaceId": "ce7ed422-2132-4a11-ab14-06c5c410f29f",
                "ExternalImageId": "example-image.jpg",
                "Confidence": 99.993408203125,
                "ImageId": "8d67061e-90d2-598f-9fbd-29c8497039c0"
            }
        }
    ],
    "UnindexedFaces": [],
    "FaceModelVersion": "3.0",
    "OrientationCorrection": "ROTATE_0"
}
```

Para obtener más información, consulte Agregar rostros a una colección en la Guía para desarrolladores de Amazon Rekognition.

- Para obtener más información sobre la API, consulta IndexFaceslа Referencia de AWS CLI comandos.

Java

SDK para Java 2.x

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.*;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
```

```
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class AddFacesToCollection {
    public static void main(String[] args) {
        final String usage = """
            Usage: <collectionId> <sourceImage> <bucketName>

            Where:
                collectionName - The name of the collection.
                sourceImage - The name of the image (for example, pic1.png).
                bucketName - The name of the S3 bucket.
            """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String collectionId = args[0];
        String sourceImage = args[1];
        String bucketName = args[2];;
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        addToCollection(rekClient, collectionId, bucketName, sourceImage);
        rekClient.close();
    }

    /**
     * Adds a face from an image to an Amazon Rekognition collection.
     *
     * @param rekClient     the Amazon Rekognition client
     * @param collectionId  the ID of the collection to add the face to
     * @param bucketName    the name of the Amazon S3 bucket containing the image
     * @param sourceImage   the name of the image file to add to the collection
     * @throws RekognitionException if there is an error while interacting with
 the Amazon Rekognition service
     */
```

```
   public static void addToCollection(RekognitionClient rekClient, String
collectionId, String bucketName, String sourceImage) {
       try {
           S3Object s3ObjectTarget = S3Object.builder()
                   .bucket(bucketName)
                   .name(sourceImage)
                   .build();

           Image targetImage = Image.builder()
                   .s3Object(s3ObjectTarget)
                   .build();

           IndexFacesRequest facesRequest = IndexFacesRequest.builder()
                   .collectionId(collectionId)
                   .image(targetImage)
                   .maxFaces(1)
                   .qualityFilter(QualityFilter.AUTO)
                   .detectionAttributes(Attribute.DEFAULT)
                   .build();

           IndexFacesResponse facesResponse =
rekClient.indexFaces(facesRequest);
           System.out.println("Results for the image");
           System.out.println("\n Faces indexed:");
           List<FaceRecord> faceRecords = facesResponse.faceRecords();
           for (FaceRecord faceRecord : faceRecords) {
               System.out.println("  Face ID: " + faceRecord.face().faceId());
               System.out.println("  Location:" +
faceRecord.faceDetail().boundingBox().toString());
           }

           List<UnindexedFace> unindexedFaces = facesResponse.unindexedFaces();
           System.out.println("Faces not indexed:");
           for (UnindexedFace unindexedFace : unindexedFaces) {
               System.out.println("  Location:" +
unindexedFace.faceDetail().boundingBox().toString());
               System.out.println("  Reasons:");
               for (Reason reason : unindexedFace.reasons()) {
                   System.out.println("Reason:  " + reason);
               }
           }

       } catch (RekognitionException e) {
           System.out.println(e.getMessage());
```

```
            System.exit(1);
        }
    }
}
```

- Para obtener más información sobre la API, consulta IndexFaceslа Referencia AWS SDK for Java 2.x de la API.

Kotlin

SDK para Kotlin

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun addToCollection(
    collectionIdVal: String?,
    sourceImage: String,
) {
    val souImage =
        Image {
            bytes = (File(sourceImage).readBytes())
        }

    val request =
        IndexFacesRequest {
            collectionId = collectionIdVal
            image = souImage
            maxFaces = 1
            qualityFilter = QualityFilter.Auto
            detectionAttributes = listOf(Attribute.Default)
        }

    RekognitionClient.fromEnvironment { region = "us-east-1" }.use { rekClient ->
        val facesResponse = rekClient.indexFaces(request)

        // Display the results.
```

```
        println("Results for the image")
        println("\n Faces indexed:")
        facesResponse.faceRecords?.forEach { faceRecord ->
            println("Face ID: ${faceRecord.face?.faceId}")
            println("Location: ${faceRecord.faceDetail?.boundingBox}")
        }

        println("Faces not indexed:")
        facesResponse.unindexedFaces?.forEach { unindexedFace ->
            println("Location: ${unindexedFace.faceDetail?.boundingBox}")
            println("Reasons:")

            unindexedFace.reasons?.forEach { reason ->
                println("Reason:  $reason")
            }
        }
    }
}
```

- Para obtener más información sobre la API, consulta IndexFaceslа referencia sobre el AWS SDK para la API de Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
class RekognitionCollection:
    """
    Encapsulates an Amazon Rekognition collection. This class is a thin wrapper
    around parts of the Boto3 Amazon Rekognition API.
    """

    def __init__(self, collection, rekognition_client):
        """
```

```
        Initializes a collection object.

        :param collection: Collection data in the format returned by a call to
                           create_collection.
        :param rekognition_client: A Boto3 Rekognition client.
        """
        self.collection_id = collection["CollectionId"]
        self.collection_arn, self.face_count, self.created =
self._unpack_collection(
            collection
        )
        self.rekognition_client = rekognition_client

    @staticmethod
    def _unpack_collection(collection):
        """
        Unpacks optional parts of a collection that can be returned by
        describe_collection.

        :param collection: The collection data.
        :return: A tuple of the data in the collection.
        """
        return (
            collection.get("CollectionArn"),
            collection.get("FaceCount", 0),
            collection.get("CreationTimestamp"),
        )


    def index_faces(self, image, max_faces):
        """
        Finds faces in the specified image, indexes them, and stores them in the
        collection.

        :param image: The image to index.
        :param max_faces: The maximum number of faces to index.
        :return: A tuple. The first element is a list of indexed faces.
                 The second element is a list of faces that couldn't be indexed.
        """
        try:
            response = self.rekognition_client.index_faces(
                CollectionId=self.collection_id,
                Image=image.image,
                ExternalImageId=image.image_name,
```

```
                MaxFaces=max_faces,
                DetectionAttributes=["ALL"],
            )
            indexed_faces = [
                RekognitionFace({**face["Face"], **face["FaceDetail"]})
                for face in response["FaceRecords"]
            ]
            unindexed_faces = [
                RekognitionFace(face["FaceDetail"])
                for face in response["UnindexedFaces"]
            ]
            logger.info(
                "Indexed %s faces in %s. Could not index %s faces.",
                len(indexed_faces),
                image.image_name,
                len(unindexed_faces),
            )
        except ClientError:
            logger.exception("Couldn't index faces in image %s.",
image.image_name)
            raise
        else:
            return indexed_faces, unindexed_faces
```

- Para obtener más información sobre la API, consulta IndexFaceslа AWS Referencia de API de SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
    TRY.
        " Create S3 object reference for the image
```

```
        DATA(lo_s3object) = NEW /aws1/cl_reks3object(
          iv_bucket = iv_s3_bucket
          iv_name = iv_s3_key ).

        " Create image object
        DATA(lo_image) = NEW /aws1/cl_rekimage(
          io_s3object = lo_s3object ).

        " Index faces in the image
        oo_result = lo_rek->indexfaces(
          iv_collectionid = iv_collection_id
          io_image = lo_image
          iv_externalimageid = iv_external_id
          iv_maxfaces = iv_max_faces ).

        DATA(lt_face_records) = oo_result->get_facerecords( ).
        DATA(lv_indexed_count) = lines( lt_face_records ).
        DATA(lv_msg2) = |{ lv_indexed_count } face(s) indexed successfully.|.
        MESSAGE lv_msg2 TYPE 'I'.
      CATCH /aws1/cx_rekresourcenotfoundex.
        MESSAGE 'Collection not found.' TYPE 'E'.
      CATCH /aws1/cx_rekinvalids3objectex.
        MESSAGE 'Invalid S3 object.' TYPE 'E'.
      CATCH /aws1/cx_rekinvalidparameterex.
        MESSAGE 'Invalid parameter value.' TYPE 'E'.
    ENDTRY.
```

- Para obtener más información sobre la API, consulte IndexFaceslа referencia sobre la API ABAP del AWS SDK para SAP.

Para obtener una lista completa de guías para desarrolladores del AWS SDK y ejemplos de código, consulteUso de Rekognition con un SDK AWS. En este tema también se incluye información sobre cómo comenzar a utilizar el SDK y detalles sobre sus versiones anteriores.

Úselo **`ListCollections`** con un AWS SDK o CLI

Los siguientes ejemplos de código muestran cómo utilizar `ListCollections`.

Para obtener más información, consulte Enumerar colecciones.

.NET

SDK para .NET

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
using System;
using System.Threading.Tasks;
using Amazon.Rekognition;
using Amazon.Rekognition.Model;

/// <summary>
/// Uses Amazon Rekognition to list the collection IDs in the
/// current account.
/// </summary>
public class ListCollections
{
    public static async Task Main()
    {
        var rekognitionClient = new AmazonRekognitionClient();

        Console.WriteLine("Listing collections");
        int limit = 10;

        var listCollectionsRequest = new ListCollectionsRequest
        {
            MaxResults = limit,
        };

        var listCollectionsResponse = new ListCollectionsResponse();

        do
        {
            if (listCollectionsResponse is not null)
            {
                listCollectionsRequest.NextToken =
listCollectionsResponse.NextToken;
            }
```

```
                listCollectionsResponse = await
rekognitionClient.ListCollectionsAsync(listCollectionsRequest);

                listCollectionsResponse.CollectionIds.ForEach(id =>
                {
                    Console.WriteLine(id);
                });
            }
            while (listCollectionsResponse.NextToken is not null);
        }
    }
```

- Para obtener más información sobre la API, consulta ListCollectionsla Referencia AWS SDK para .NET de la API.

CLI

AWS CLI

Creación de una lista de las colecciones disponibles

El siguiente `list-collections` comando muestra las colecciones disponibles en la AWS cuenta.

```
aws rekognition list-collections
```

Salida:

```
{
    "FaceModelVersions": [
        "2.0",
        "3.0",
        "3.0",
        "3.0",
        "4.0",
        "1.0",
        "3.0",
        "4.0",
```

```
        "4.0",
        "4.0"
    ],
    "CollectionIds": [
        "MyCollection1",
        "MyCollection2",
        "MyCollection3",
        "MyCollection4",
        "MyCollection5",
        "MyCollection6",
        "MyCollection7",
        "MyCollection8",
        "MyCollection9",
        "MyCollection10"
    ]
}
```

Para obtener más información, consulte Listado de colecciones en la Guía para desarrolladores de Amazon Rekognition.

- Para obtener más información sobre la API, consulte ListCollectionsla Referencia de AWS CLI comandos.

Java

SDK para Java 2.x

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.ListCollectionsRequest;
import software.amazon.awssdk.services.rekognition.model.ListCollectionsResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
```

```
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class ListCollections {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        System.out.println("Listing collections");
        listAllCollections(rekClient);
        rekClient.close();
    }

    public static void listAllCollections(RekognitionClient rekClient) {
        try {
            ListCollectionsRequest listCollectionsRequest =
 ListCollectionsRequest.builder()
                    .maxResults(10)
                    .build();

            ListCollectionsResponse response =
 rekClient.listCollections(listCollectionsRequest);
            List<String> collectionIds = response.collectionIds();
            for (String resultId : collectionIds) {
                System.out.println(resultId);
            }

        } catch (RekognitionException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Para obtener más información sobre la API, consulta ListCollectionsla Referencia AWS SDK for Java 2.x de la API.

Kotlin

SDK para Kotlin

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun listAllCollections() {
    val request =
        ListCollectionsRequest {
            maxResults = 10
        }

    RekognitionClient.fromEnvironment { region = "us-east-1" }.use { rekClient ->
        val response = rekClient.listCollections(request)
        response.collectionIds?.forEach { resultId ->
            println(resultId)
        }
    }
}
```

- Para obtener más información sobre la API, consulta ListCollectionsla referencia sobre el AWS SDK para la API de Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
class RekognitionCollectionManager:
```

```
    """
    Encapsulates Amazon Rekognition collection management functions.
    This class is a thin wrapper around parts of the Boto3 Amazon Rekognition
API.
    """

    def __init__(self, rekognition_client):
        """
        Initializes the collection manager object.

        :param rekognition_client: A Boto3 Rekognition client.
        """
        self.rekognition_client = rekognition_client


    def list_collections(self, max_results):
        """
        Lists collections for the current account.

        :param max_results: The maximum number of collections to return.
        :return: The list of collections for the current account.
        """
        try:
            response =
self.rekognition_client.list_collections(MaxResults=max_results)
            collections = [
                RekognitionCollection({"CollectionId": col_id},
self.rekognition_client)
                for col_id in response["CollectionIds"]
            ]
        except ClientError:
            logger.exception("Couldn't list collections.")
            raise
        else:
            return collections
```

- Para obtener más información sobre la API, consulta ListCollectionsla AWS Referencia de API de SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
TRY.
    oo_result = lo_rek->listcollections(
      iv_maxresults = iv_max_results ).

    DATA(lt_collection_ids) = oo_result->get_collectionids( ).
    DATA(lv_coll_count) = lines( lt_collection_ids ).
    DATA(lv_msg7) = |{ lv_coll_count } collection(s) found.|.
    MESSAGE lv_msg7 TYPE 'I'.
  CATCH /aws1/cx_rekinvalidparameterex.
    MESSAGE 'Invalid parameter value.' TYPE 'E'.
ENDTRY.
```

- Para obtener más información sobre la API, consulte ListCollectionsla referencia sobre la API ABAP del AWS SDK para SAP.

Para obtener una lista completa de guías para desarrolladores del AWS SDK y ejemplos de código, consulteUso de Rekognition con un SDK AWS. En este tema también se incluye información sobre cómo comenzar a utilizar el SDK y detalles sobre sus versiones anteriores.

Úselo **`ListFaces`** con un AWS SDK o CLI

Los siguientes ejemplos de código muestran cómo utilizar `ListFaces`.

Para obtener más información, consulte Enumerar rostros en una colección.

.NET

SDK para .NET

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
using System;
using System.Threading.Tasks;
using Amazon.Rekognition;
using Amazon.Rekognition.Model;

/// <summary>
/// Uses the Amazon Rekognition Service to retrieve the list of faces
/// stored in a collection.
/// </summary>
public class ListFaces
{
    public static async Task Main()
    {
        string collectionId = "MyCollection2";

        var rekognitionClient = new AmazonRekognitionClient();

        var listFacesResponse = new ListFacesResponse();
        Console.WriteLine($"Faces in collection {collectionId}");

        var listFacesRequest = new ListFacesRequest
        {
            CollectionId = collectionId,
            MaxResults = 1,
        };

        do
        {
            listFacesResponse = await
rekognitionClient.ListFacesAsync(listFacesRequest);
            listFacesResponse.Faces.ForEach(face =>
            {
```

```
                    Console.WriteLine(face.FaceId);
                });

                listFacesRequest.NextToken = listFacesResponse.NextToken;
            }
            while (!string.IsNullOrEmpty(listFacesResponse.NextToken));
        }
    }
```

- Para obtener más información sobre la API, consulta ListFaceslla Referencia AWS SDK para .NET de la API.

CLI

AWS CLI

Creación de una lista de rostros en una colección

El siguiente comando `list-faces` muestra los rostros de la colección especificada.

```
aws rekognition list-faces \
    --collection-id MyCollection
```

Salida:

```
{
    "FaceModelVersion": "3.0",
    "Faces": [
        {
            "BoundingBox": {
                "Width": 0.5216310024261475,
                "Top": 0.3256250023841858,
                "Left": 0.13394300639629364,
                "Height": 0.3918749988079071
            },
            "FaceId": "0040279c-0178-436e-b70a-e61b074e96b0",
            "ExternalImageId": "image1.jpg",
            "Confidence": 100.0,
            "ImageId": "f976e487-3719-5e2d-be8b-ea2724c26991"
        },
```

```
        {
            "BoundingBox": {
                "Width": 0.5074880123138428,
                "Top": 0.3774999976158142,
                "Left": 0.18302799761295319,
                "Height": 0.3812499940395355
            },
            "FaceId": "086261e8-6deb-4bc0-ac73-ab22323cc38d",
            "ExternalImageId": "image2.jpg",
            "Confidence": 99.99930572509766,
            "ImageId": "ae1593b0-a8f6-5e24-a306-abf529e276fa"
        },
        {
            "BoundingBox": {
                "Width": 0.5574039816856384,
                "Top": 0.37187498807907104,
                "Left": 0.14559100568294525,
                "Height": 0.4181250035762787
            },
            "FaceId": "11c4bd3c-19c5-4eb8-aecc-24feb93a26e1",
            "ExternalImageId": "image3.jpg",
            "Confidence": 99.99960327148438,
            "ImageId": "80739b4d-883f-5b78-97cf-5124038e26b9"
        },
        {
            "BoundingBox": {
                "Width": 0.18562500178813934,
                "Top": 0.1618019938468933,
                "Left": 0.5575000047683716,
                "Height": 0.24770599603652954
            },
            "FaceId": "13692fe4-990a-4679-b14a-5ac23d135eab",
            "ExternalImageId": "image4.jpg",
            "Confidence": 99.99340057373047,
            "ImageId": "8df18239-9ad1-5acd-a46a-6581ff98f51b"
        },
        {
            "BoundingBox": {
                "Width": 0.5307819843292236,
                "Top": 0.2862499952316284,
                "Left": 0.1564060002565384,
                "Height": 0.3987500071525574
            },
            "FaceId": "2eb5f3fd-e2a9-4b1c-a89f-afa0a518fe06",
```

```
            "ExternalImageId": "image5.jpg",
            "Confidence": 99.99970245361328,
            "ImageId": "3c314792-197d-528d-bbb6-798ed012c150"
        },
        {
            "BoundingBox": {
                "Width": 0.5773710012435913,
                "Top": 0.34437501430511475,
                "Left": 0.12396000325679779,
                "Height": 0.4337500035762787
            },
            "FaceId": "57189455-42b0-4839-a86c-abda48b13174",
            "ExternalImageId": "image6.jpg",
            "Confidence": 100.0,
            "ImageId": "0aff2f37-e7a2-5dbc-a3a3-4ef6ec18eaa0"
        },
        {
            "BoundingBox": {
                "Width": 0.5349419713020325,
                "Top": 0.29124999046325684,
                "Left": 0.16389399766921997,
                "Height": 0.40187498927116394
            },
            "FaceId": "745f7509-b1fa-44e0-8b95-367b1359638a",
            "ExternalImageId": "image7.jpg",
            "Confidence": 99.99979400634766,
            "ImageId": "67a34327-48d1-5179-b042-01e52ccfeada"
        },
        {
            "BoundingBox": {
                "Width": 0.41499999165534973,
                "Top": 0.09187500178813934,
                "Left": 0.28083300590515137,
                "Height": 0.3112500011920929
            },
            "FaceId": "8d3cfc70-4ba8-4b36-9644-90fba29c2dac",
            "ExternalImageId": "image8.jpg",
            "Confidence": 99.99769592285156,
            "ImageId": "a294da46-2cb1-5cc4-9045-61d7ca567662"
        },
        {
            "BoundingBox": {
                "Width": 0.48166701197624207,
                "Top": 0.20999999344348907,
```

```
                "Left": 0.21250000596046448,
                "Height": 0.36125001311302185
            },
            "FaceId": "bd4ceb4d-9acc-4ab7-8ef8-1c2d2ba0a66a",
            "ExternalImageId": "image9.jpg",
            "Confidence": 99.99949645996094,
            "ImageId": "5e1a7588-e5a0-5ee3-bd00-c642518dfe3a"
        },
        {
            "BoundingBox": {
                "Width": 0.18562500178813934,
                "Top": 0.1618019938468933,
                "Left": 0.5575000047683716,
                "Height": 0.24770599603652954
            },
            "FaceId": "ce7ed422-2132-4a11-ab14-06c5c410f29f",
            "ExternalImageId": "image10.jpg",
            "Confidence": 99.99340057373047,
            "ImageId": "8d67061e-90d2-598f-9fbd-29c8497039c0"
        }
    ]
}
```

Para obtener más información, consulte Listado de rostros en una colección en la Guía para desarrolladores de Amazon Rekognition.

- Para obtener más información sobre la API, consulta ListFaceslа Referencia de AWS CLI comandos.

Java

SDK para Java 2.x

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.Face;
```

```
import software.amazon.awssdk.services.rekognition.model.ListFacesRequest;
import software.amazon.awssdk.services.rekognition.model.ListFacesResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class ListFacesInCollection {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <collectionId>

                Where:
                   collectionId - The name of the collection.\s
                """;

        if (args.length < 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String collectionId = args[0];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        System.out.println("Faces in collection " + collectionId);
        listFacesCollection(rekClient, collectionId);
        rekClient.close();
    }

    public static void listFacesCollection(RekognitionClient rekClient, String
 collectionId) {
        try {
            ListFacesRequest facesRequest = ListFacesRequest.builder()
```

```
                    .collectionId(collectionId)
                    .maxResults(10)
                    .build();

            ListFacesResponse facesResponse = rekClient.listFaces(facesRequest);
            List<Face> faces = facesResponse.faces();
            for (Face face : faces) {
                System.out.println("Confidence level there is a face: " +
 face.confidence());
                System.out.println("The face Id value is " + face.faceId());
            }

        } catch (RekognitionException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Para obtener más información sobre la API, consulta ListFaceslla Referencia AWS SDK for Java 2.x de la API.

Kotlin

SDK para Kotlin

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun listFacesCollection(collectionIdVal: String?) {
    val request =
        ListFacesRequest {
            collectionId = collectionIdVal
            maxResults = 10
        }

    RekognitionClient.fromEnvironment { region = "us-east-1" }.use { rekClient ->
```

```
        val response = rekClient.listFaces(request)
        response.faces?.forEach { face ->
            println("Confidence level there is a face: ${face.confidence}")
            println("The face Id value is ${face.faceId}")
        }
    }
}
```

- Para obtener más información sobre la API, consulta ListFaceslla referencia sobre el AWS SDK para la API de Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
class RekognitionCollection:
    """
    Encapsulates an Amazon Rekognition collection. This class is a thin wrapper
    around parts of the Boto3 Amazon Rekognition API.
    """

    def __init__(self, collection, rekognition_client):
        """
        Initializes a collection object.

        :param collection: Collection data in the format returned by a call to
                           create_collection.
        :param rekognition_client: A Boto3 Rekognition client.
        """
        self.collection_id = collection["CollectionId"]
        self.collection_arn, self.face_count, self.created =
 self._unpack_collection(
            collection
        )
```

```
        self.rekognition_client = rekognition_client

    @staticmethod
    def _unpack_collection(collection):
        """
        Unpacks optional parts of a collection that can be returned by
        describe_collection.

        :param collection: The collection data.
        :return: A tuple of the data in the collection.
        """
        return (
            collection.get("CollectionArn"),
            collection.get("FaceCount", 0),
            collection.get("CreationTimestamp"),
        )


    def list_faces(self, max_results):
        """
        Lists the faces currently indexed in the collection.

        :param max_results: The maximum number of faces to return.
        :return: The list of faces in the collection.
        """
        try:
            response = self.rekognition_client.list_faces(
                CollectionId=self.collection_id, MaxResults=max_results
            )
            faces = [RekognitionFace(face) for face in response["Faces"]]
            logger.info(
                "Found %s faces in collection %s.", len(faces),
self.collection_id
            )
        except ClientError:
            logger.exception(
                "Couldn't list faces in collection %s.", self.collection_id
            )
            raise
        else:
            return faces
```

- Para obtener más información sobre la API, consulta ListFaceslaAWS Referencia de API de SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
TRY.
    oo_result = lo_rek->listfaces(
      iv_collectionid = iv_collection_id
      iv_maxresults = iv_max_results ).

    DATA(lt_faces) = oo_result->get_faces( ).
    DATA(lv_face_count2) = lines( lt_faces ).
    DATA(lv_msg3) = |{ lv_face_count2 } face(s) found in collection.|.
    MESSAGE lv_msg3 TYPE 'I'.
  CATCH /aws1/cx_rekresourcenotfoundex.
    MESSAGE 'Collection not found.' TYPE 'E'.
  CATCH /aws1/cx_rekinvalidparameterex.
    MESSAGE 'Invalid parameter value.' TYPE 'E'.
ENDTRY.
```

- Para obtener más información sobre la API, consulte ListFaceslareferencia sobre la API ABAP del AWS SDK para SAP.

Para obtener una lista completa de guías para desarrolladores del AWS SDK y ejemplos de código, consulteUso de Rekognition con un SDK AWS. En este tema también se incluye información sobre cómo comenzar a utilizar el SDK y detalles sobre sus versiones anteriores.

Úselo **`RecognizeCelebrities`** con un AWS SDK o CLI

Los siguientes ejemplos de código muestran cómo utilizar `RecognizeCelebrities`.

Para obtener más información, consulte Reconocimiento de famosos en una imagen.

.NET

SDK para .NET

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
using System;
using System.IO;
using System.Threading.Tasks;
using Amazon.Rekognition;
using Amazon.Rekognition.Model;

/// <summary>
/// Shows how to use Amazon Rekognition to identify celebrities in a photo.
/// </summary>
public class CelebritiesInImage
{
    public static async Task Main(string[] args)
    {
        string photo = "moviestars.jpg";

        var rekognitionClient = new AmazonRekognitionClient();

        var recognizeCelebritiesRequest = new RecognizeCelebritiesRequest();

        var img = new Amazon.Rekognition.Model.Image();
        byte[] data = null;
        try
        {
            using var fs = new FileStream(photo, FileMode.Open,
FileAccess.Read);
            data = new byte[fs.Length];
            fs.Read(data, 0, (int)fs.Length);
        }
        catch (Exception)
        {
```

```
                Console.WriteLine($"Failed to load file {photo}");
                return;
            }

            img.Bytes = new MemoryStream(data);
            recognizeCelebritiesRequest.Image = img;

            Console.WriteLine($"Looking for celebrities in image {photo}\n");

            var recognizeCelebritiesResponse = await
rekognitionClient.RecognizeCelebritiesAsync(recognizeCelebritiesRequest);


Console.WriteLine($"{recognizeCelebritiesResponse.CelebrityFaces.Count}
celebrity(s) were recognized.\n");
            recognizeCelebritiesResponse.CelebrityFaces.ForEach(celeb =>
            {
                Console.WriteLine($"Celebrity recognized: {celeb.Name}");
                Console.WriteLine($"Celebrity ID: {celeb.Id}");
                BoundingBox boundingBox = celeb.Face.BoundingBox;
                Console.WriteLine($"position: {boundingBox.Left}
{boundingBox.Top}");
                Console.WriteLine("Further information (if available):");
                celeb.Urls.ForEach(url =>
                {
                    Console.WriteLine(url);
                });
            });


Console.WriteLine($"{recognizeCelebritiesResponse.UnrecognizedFaces.Count}
face(s) were unrecognized.");
        }
    }
```

- Para obtener más información sobre la API, consulta RecognizeCelebritiesla Referencia AWS SDK para .NET de la API.

CLI

AWS CLI

Reconocimiento de famosos en una imagen

El siguiente comando `recognize-celebrities` reconoce a famosos en la imagen especificada almacenada en un bucket de Amazon S3:

```
aws rekognition recognize-celebrities \
    --image "S3Object={Bucket=MyImageS3Bucket,Name=moviestars.jpg}"
```

Salida:

```
{
    "UnrecognizedFaces": [
        {
            "BoundingBox": {
                "Width": 0.14416666328907013,
                "Top": 0.07777778059244156,
                "Left": 0.625,
                "Height": 0.2746031880378723
            },
            "Confidence": 99.9990234375,
            "Pose": {
                "Yaw": 10.80408763885498,
                "Roll": -12.761146545410156,
                "Pitch": 10.96889877319336
            },
            "Quality": {
                "Sharpness": 94.1185531616211,
                "Brightness": 79.18367004394531
            },
            "Landmarks": [
                {
                    "Y": 0.18220913410186768,
                    "X": 0.6702951788902283,
                    "Type": "eyeLeft"
                },
                {
                    "Y": 0.16337193548679352,
                    "X": 0.7188183665275574,
                    "Type": "eyeRight"
```

```
                },
                {
                    "Y": 0.20739148557186127,
                    "X": 0.7055801749229431,
                    "Type": "nose"
                },
                {
                    "Y": 0.2889308035373688,
                    "X": 0.687512218952179,
                    "Type": "mouthLeft"
                },
                {
                    "Y": 0.2706988751888275,
                    "X": 0.7250053286552429,
                    "Type": "mouthRight"
                }
            ]
        }
    ],
    "CelebrityFaces": [
        {
            "MatchConfidence": 100.0,
            "Face": {
                "BoundingBox": {
                    "Width": 0.14000000059604645,
                    "Top": 0.1190476194024086,
                    "Left": 0.82833331823349,
                    "Height": 0.2666666805744171
                },
                "Confidence": 99.99359130859375,
                "Pose": {
                    "Yaw": -10.509642601013184,
                    "Roll": -14.51749324798584,
                    "Pitch": 13.799399375915527
                },
                "Quality": {
                    "Sharpness": 78.74752044677734,
                    "Brightness": 42.201324462890625
                },
                "Landmarks": [
                    {
                        "Y": 0.2290833294391632,
                        "X": 0.8709492087364197,
                        "Type": "eyeLeft"
```

```
                    },
                    {
                        "Y": 0.20639978349208832,
                        "X": 0.9153988361358643,
                        "Type": "eyeRight"
                    },
                    {
                        "Y": 0.25417643785476685,
                        "X": 0.8907724022865295,
                        "Type": "nose"
                    },
                    {
                        "Y": 0.32729196548461914,
                        "X": 0.8876466155052185,
                        "Type": "mouthLeft"
                    },
                    {
                        "Y": 0.3115464746952057,
                        "X": 0.9238573312759399,
                        "Type": "mouthRight"
                    }
                ]
            },
            "Name": "Celeb A",
            "Urls": [
                "www.imdb.com/name/aaaaaaaaa"
            ],
            "Id": "1111111"
        },
        {
            "MatchConfidence": 97.0,
            "Face": {
                "BoundingBox": {
                    "Width": 0.13333334028720856,
                    "Top": 0.24920634925365448,
                    "Left": 0.4449999928474426,
                    "Height": 0.2539682686328888
                },
                "Confidence": 99.99979400634766,
                "Pose": {
                    "Yaw": 6.557040691375732,
                    "Roll": -7.316643714904785,
                    "Pitch": 9.272967338562012
                },
```

```
                "Quality": {
                    "Sharpness": 83.23492431640625,
                    "Brightness": 78.83267974853516
                },
                "Landmarks": [
                    {
                        "Y": 0.3625510632991791,
                        "X": 0.48898839950561523,
                        "Type": "eyeLeft"
                    },
                    {
                        "Y": 0.35366007685661316,
                        "X": 0.5313721299171448,
                        "Type": "eyeRight"
                    },
                    {
                        "Y": 0.3894785940647125,
                        "X": 0.5173314809799194,
                        "Type": "nose"
                    },
                    {
                        "Y": 0.44889405369758606,
                        "X": 0.5020005702972412,
                        "Type": "mouthLeft"
                    },
                    {
                        "Y": 0.4408611059188843,
                        "X": 0.5351271629333496,
                        "Type": "mouthRight"
                    }
                ]
            },
            "Name": "Celeb B",
            "Urls": [
                "www.imdb.com/name/bbbbbbbbb"
            ],
            "Id": "2222222"
        },
        {
            "MatchConfidence": 100.0,
            "Face": {
                "BoundingBox": {
                    "Width": 0.12416666746139526,
                    "Top": 0.2968254089355469,
```

```
                    "Left": 0.2150000035762787,
                    "Height": 0.23650793731212616
                },
                "Confidence": 99.99958801269531,
                "Pose": {
                    "Yaw": 7.801797866821289,
                    "Roll": -8.326810836791992,
                    "Pitch": 7.844768047332764
                },
                "Quality": {
                    "Sharpness": 86.93206024169922,
                    "Brightness": 79.81291198730469
                },
                "Landmarks": [
                    {
                        "Y": 0.4027804136276245,
                        "X": 0.2575301229953766,
                        "Type": "eyeLeft"
                    },
                    {
                        "Y": 0.3934555947780609,
                        "X": 0.2956969439983368,
                        "Type": "eyeRight"
                    },
                    {
                        "Y": 0.4309830069541931,
                        "X": 0.2837020754814148,
                        "Type": "nose"
                    },
                    {
                        "Y": 0.48186683654785156,
                        "X": 0.26812544465065,
                        "Type": "mouthLeft"
                    },
                    {
                        "Y": 0.47338807582855225,
                        "X": 0.29905644059181213,
                        "Type": "mouthRight"
                    }
                ]
            },
            "Name": "Celeb C",
            "Urls": [
                "www.imdb.com/name/ccccccccc"
```

```
            ],
            "Id": "3333333"
        },
        {
            "MatchConfidence": 97.0,
            "Face": {
                "BoundingBox": {
                    "Width": 0.11916666477918625,
                    "Top": 0.3698412775993347,
                    "Left": 0.008333333767950535,
                    "Height": 0.22698412835597992
                },
                "Confidence": 99.99999237060547,
                "Pose": {
                    "Yaw": 16.38478660583496,
                    "Roll": -1.0260354280471802,
                    "Pitch": 5.975185394287109
                },
                "Quality": {
                    "Sharpness": 83.23492431640625,
                    "Brightness": 61.408443450927734
                },
                "Landmarks": [
                    {
                        "Y": 0.4632347822189331,
                        "X": 0.049406956881284714,
                        "Type": "eyeLeft"
                    },
                    {
                        "Y": 0.46388113498687744,
                        "X": 0.08722897619009018,
                        "Type": "eyeRight"
                    },
                    {
                        "Y": 0.5020678639411926,
                        "X": 0.0758260041475296,
                        "Type": "nose"
                    },
                    {
                        "Y": 0.544157862663269,
                        "X": 0.054029736667871475,
                        "Type": "mouthLeft"
                    },
                    {
```

```
                        "Y": 0.5463630557060242,
                        "X": 0.08464983850717545,
                        "Type": "mouthRight"
                    }
                ]
            },
            "Name": "Celeb D",
            "Urls": [
                "www.imdb.com/name/ddddddddd"
            ],
            "Id": "4444444"
        }
    ]
}
```

Para obtener más información, consulte Reconocimiento de famosos en una imagen en la Guía para desarrolladores de Amazon Rekognition.

- Para obtener más información sobre la API, consulta RecognizeCelebritiesla Referencia de AWS CLI comandos.

Java

SDK para Java 2.x

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.core.SdkBytes;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;
import java.util.List;

import software.amazon.awssdk.services.rekognition.model.*;

/**
```

```
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class RecognizeCelebrities {
    public static void main(String[] args) {
        final String usage = """
                Usage:   <bucketName> <sourceImage>

                Where:
                   bucketName - The name of the S3 bucket where the images are
 stored.
                   sourceImage - The path to the image (for example, C:\\AWS\
\pic1.png).\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
       }

        String bucketName = args[0];;
        String sourceImage = args[1];
        Region region = Region.US_WEST_2;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        System.out.println("Locating celebrities in " + sourceImage);
        recognizeAllCelebrities(rekClient, bucketName, sourceImage);
        rekClient.close();
    }

    /**
     * Recognizes all celebrities in an image stored in an Amazon S3 bucket.
     *
     * @param rekClient    the Amazon Rekognition client used to perform the
 celebrity recognition operation
     * @param bucketName   the name of the Amazon S3 bucket where the source
 image is stored
```

```
     * @param sourceImage  the name of the source image file stored in the Amazon
 S3 bucket
     */
    public static void recognizeAllCelebrities(RekognitionClient rekClient,
 String bucketName, String sourceImage) {
        try {
            S3Object s3ObjectTarget = S3Object.builder()
                .bucket(bucketName)
                .name(sourceImage)
                .build();

            Image souImage = Image.builder()
                    .s3Object(s3ObjectTarget)
                    .build();

            RecognizeCelebritiesRequest request =
 RecognizeCelebritiesRequest.builder()
                    .image(souImage)
                    .build();

            RecognizeCelebritiesResponse result =
 rekClient.recognizeCelebrities(request);
            List<Celebrity> celebs = result.celebrityFaces();
            System.out.println(celebs.size() + " celebrity(s) were recognized.
\n");
            for (Celebrity celebrity : celebs) {
                System.out.println("Celebrity recognized: " + celebrity.name());
                System.out.println("Celebrity ID: " + celebrity.id());

                System.out.println("Further information (if available):");
                for (String url : celebrity.urls()) {
                    System.out.println(url);
                }
                System.out.println();
            }
            System.out.println(result.unrecognizedFaces().size() + " face(s) were
 unrecognized.");

        } catch (RekognitionException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Para obtener más información sobre la API, consulta RecognizeCelebritiesla Referencia AWS SDK for Java 2.x de la API.

Kotlin

SDK para Kotlin

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun recognizeAllCelebrities(sourceImage: String?) {
    val souImage =
        Image {
            bytes = (File(sourceImage).readBytes())
        }

    val request =
        RecognizeCelebritiesRequest {
            image = souImage
        }

    RekognitionClient.fromEnvironment { region = "us-east-1" }.use { rekClient ->
        val response = rekClient.recognizeCelebrities(request)
        response.celebrityFaces?.forEach { celebrity ->
            println("Celebrity recognized: ${celebrity.name}")
            println("Celebrity ID:${celebrity.id}")
            println("Further information (if available):")
            celebrity.urls?.forEach { url ->
                println(url)
            }
        }
        println("${response.unrecognizedFaces?.size} face(s) were unrecognized.")
    }
}
```

- Para obtener más información sobre la API, consulta RecognizeCelebritiesla referencia sobre el AWS SDK para la API de Kotlin.

Python

SDK para Python (Boto3)

> **Note**
>
> Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
class RekognitionImage:
    """
    Encapsulates an Amazon Rekognition image. This class is a thin wrapper
    around parts of the Boto3 Amazon Rekognition API.
    """

    def __init__(self, image, image_name, rekognition_client):
        """
        Initializes the image object.

        :param image: Data that defines the image, either the image bytes or
                      an Amazon S3 bucket and object key.
        :param image_name: The name of the image.
        :param rekognition_client: A Boto3 Rekognition client.
        """
        self.image = image
        self.image_name = image_name
        self.rekognition_client = rekognition_client


    def recognize_celebrities(self):
        """
        Detects celebrities in the image.

        :return: A tuple. The first element is the list of celebrities found in
                 the image. The second element is the list of faces that were
                 detected but did not match any known celebrities.
        """
```

```
        try:
            response =
self.rekognition_client.recognize_celebrities(Image=self.image)
            celebrities = [
                RekognitionCelebrity(celeb) for celeb in
response["CelebrityFaces"]
            ]
            other_faces = [
                RekognitionFace(face) for face in response["UnrecognizedFaces"]
            ]
            logger.info(
                "Found %s celebrities and %s other faces in %s.",
                len(celebrities),
                len(other_faces),
                self.image_name,
            )
        except ClientError:
            logger.exception("Couldn't detect celebrities in %s.",
self.image_name)
            raise
        else:
            return celebrities, other_faces
```

- Para obtener más información sobre la API, consulta RecognizeCelebritiesla AWS Referencia de API de SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
    TRY.
        " Create S3 object reference for the image
```

```
        DATA(lo_s3object) = NEW /aws1/cl_reks3object(
          iv_bucket = iv_s3_bucket
          iv_name = iv_s3_key ).

        " Create image object
        DATA(lo_image) = NEW /aws1/cl_rekimage(
          io_s3object = lo_s3object ).

        " Recognize celebrities
        oo_result = lo_rek->recognizecelebrities(
          io_image = lo_image ).

        DATA(lt_celebrity_faces) = oo_result->get_celebrityfaces( ).
        DATA(lv_celeb_count) = lines( lt_celebrity_faces ).
        DATA(lv_msg12) = |{ lv_celeb_count } celebrity/celebrities recognized.|.
        MESSAGE lv_msg12 TYPE 'I'.
      CATCH /aws1/cx_rekinvalids3objectex.
        MESSAGE 'Invalid S3 object.' TYPE 'E'.
      CATCH /aws1/cx_rekinvalidparameterex.
        MESSAGE 'Invalid parameter value.' TYPE 'E'.
    ENDTRY.
```

- Para obtener más información sobre la API, consulte RecognizeCelebritiesla referencia sobre la API ABAP del AWS SDK para SAP.

Para obtener una lista completa de guías para desarrolladores del AWS SDK y ejemplos de código, consulteUso de Rekognition con un SDK AWS. En este tema también se incluye información sobre cómo comenzar a utilizar el SDK y detalles sobre sus versiones anteriores.

Úselo **`SearchFaces`** con un AWS SDK o CLI

Los siguientes ejemplos de código muestran cómo utilizar `SearchFaces`.

Para obtener más información, consulte Búsqueda de un rostro (ID de rostro).

.NET

SDK para .NET

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
    using System;
    using System.Threading.Tasks;
    using Amazon.Rekognition;
    using Amazon.Rekognition.Model;

    /// <summary>
    /// Uses the Amazon Rekognition Service to find faces in an image that
    /// match the face Id provided in the method request.
    /// </summary>
    public class SearchFacesMatchingId
    {
        public static async Task Main()
        {
            string collectionId = "MyCollection";
            string faceId = "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx";

            var rekognitionClient = new AmazonRekognitionClient();

            // Search collection for faces matching the face id.
            var searchFacesRequest = new SearchFacesRequest
            {
                CollectionId = collectionId,
                FaceId = faceId,
                FaceMatchThreshold = 70F,
                MaxFaces = 2,
            };

            SearchFacesResponse searchFacesResponse = await
rekognitionClient.SearchFacesAsync(searchFacesRequest);

            Console.WriteLine("Face matching faceId " + faceId);
```

```
            Console.WriteLine("Matche(s): ");
            searchFacesResponse.FaceMatches.ForEach(face =>
            {
                Console.WriteLine($"FaceId: {face.Face.FaceId} Similarity:
 {face.Similarity}");
            });
        }
    }
```

- Para obtener más información sobre la API, consulta SearchFaceslа Referencia AWS SDK para .NET de la API.

CLI

AWS CLI

Búsqueda de rostros en una colección que coincidan con un ID de rostro.

El siguiente comando `search-faces` busca rostros en una colección que coincidan con el ID de rostro especificado.

```
aws rekognition search-faces \
    --face-id 8d3cfc70-4ba8-4b36-9644-90fba29c2dac \
    --collection-id MyCollection
```

Salida:

```
{
    "SearchedFaceId": "8d3cfc70-4ba8-4b36-9644-90fba29c2dac",
    "FaceModelVersion": "3.0",
    "FaceMatches": [
        {
            "Face": {
                "BoundingBox": {
                    "Width": 0.48166701197624207,
                    "Top": 0.20999999344348907,
                    "Left": 0.21250000596046448,
                    "Height": 0.36125001311302185
                },
                "FaceId": "bd4ceb4d-9acc-4ab7-8ef8-1c2d2ba0a66a",
```

```
                "ExternalImageId": "image1.jpg",
                "Confidence": 99.99949645996094,
                "ImageId": "5e1a7588-e5a0-5ee3-bd00-c642518dfe3a"
            },
            "Similarity": 99.30997467041016
        },
        {
            "Face": {
                "BoundingBox": {
                    "Width": 0.18562500178813934,
                    "Top": 0.1618019938468933,
                    "Left": 0.5575000047683716,
                    "Height": 0.24770599603652954
                },
                "FaceId": "ce7ed422-2132-4a11-ab14-06c5c410f29f",
                "ExternalImageId": "example-image.jpg",
                "Confidence": 99.99340057373047,
                "ImageId": "8d67061e-90d2-598f-9fbd-29c8497039c0"
            },
            "Similarity": 99.24862670898438
        },
        {
            "Face": {
                "BoundingBox": {
                    "Width": 0.18562500178813934,
                    "Top": 0.1618019938468933,
                    "Left": 0.5575000047683716,
                    "Height": 0.24770599603652954
                },
                "FaceId": "13692fe4-990a-4679-b14a-5ac23d135eab",
                "ExternalImageId": "image3.jpg",
                "Confidence": 99.99340057373047,
                "ImageId": "8df18239-9ad1-5acd-a46a-6581ff98f51b"
            },
            "Similarity": 99.24862670898438
        },
        {
            "Face": {
                "BoundingBox": {
                    "Width": 0.5349419713020325,
                    "Top": 0.29124999046325684,
                    "Left": 0.16389399766921997,
                    "Height": 0.40187498927116394
                },
```

```
                "FaceId": "745f7509-b1fa-44e0-8b95-367b1359638a",
                "ExternalImageId": "image9.jpg",
                "Confidence": 99.99979400634766,
                "ImageId": "67a34327-48d1-5179-b042-01e52ccfeada"
            },
            "Similarity": 96.73158264160156
        },
        {
            "Face": {
                "BoundingBox": {
                    "Width": 0.5307819843292236,
                    "Top": 0.2862499952316284,
                    "Left": 0.1564060002565384,
                    "Height": 0.3987500071525574
                },
                "FaceId": "2eb5f3fd-e2a9-4b1c-a89f-afa0a518fe06",
                "ExternalImageId": "image10.jpg",
                "Confidence": 99.99970245361328,
                "ImageId": "3c314792-197d-528d-bbb6-798ed012c150"
            },
            "Similarity": 96.48291015625
        },
        {
            "Face": {
                "BoundingBox": {
                    "Width": 0.5074880123138428,
                    "Top": 0.3774999976158142,
                    "Left": 0.18302799761295319,
                    "Height": 0.3812499940395355
                },
                "FaceId": "086261e8-6deb-4bc0-ac73-ab22323cc38d",
                "ExternalImageId": "image6.jpg",
                "Confidence": 99.99930572509766,
                "ImageId": "ae1593b0-a8f6-5e24-a306-abf529e276fa"
            },
            "Similarity": 96.43287658691406
        },
        {
            "Face": {
                "BoundingBox": {
                    "Width": 0.5574039816856384,
                    "Top": 0.37187498807907104,
                    "Left": 0.14559100568294525,
                    "Height": 0.4181250035762787
```

```
                },
                "FaceId": "11c4bd3c-19c5-4eb8-aecc-24feb93a26e1",
                "ExternalImageId": "image5.jpg",
                "Confidence": 99.99960327148438,
                "ImageId": "80739b4d-883f-5b78-97cf-5124038e26b9"
            },
            "Similarity": 95.25305938720703
        },
        {
            "Face": {
                "BoundingBox": {
                    "Width": 0.5773710012435913,
                    "Top": 0.34437501430511475,
                    "Left": 0.12396000325679779,
                    "Height": 0.4337500035762787
                },
                "FaceId": "57189455-42b0-4839-a86c-abda48b13174",
                "ExternalImageId": "image8.jpg",
                "Confidence": 100.0,
                "ImageId": "0aff2f37-e7a2-5dbc-a3a3-4ef6ec18eaa0"
            },
            "Similarity": 95.22837829589844
        }
    ]
}
```


Para obtener más información, consulte Búsqueda de un rostro utilizando su ID de rostro en la Guía para desarrolladores de Amazon Rekognition.

- Para obtener más información sobre la API, consulta SearchFaceslа Referencia de AWS CLI comandos.

Java

SDK para Java 2.x

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import
 software.amazon.awssdk.services.rekognition.model.SearchFacesByImageRequest;
import software.amazon.awssdk.services.rekognition.model.Image;
import
 software.amazon.awssdk.services.rekognition.model.SearchFacesByImageResponse;
import software.amazon.awssdk.services.rekognition.model.FaceMatch;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class SearchFaceMatchingImageCollection {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <collectionId> <sourceImage>

                Where:
                   collectionId - The id of the collection. \s
                   sourceImage - The path to the image (for example, C:\\AWS\
\pic1.png).\s

                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }
```

```
        String collectionId = args[0];
        String sourceImage = args[1];
        Region region = Region.US_WEST_2;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        System.out.println("Searching for a face in a collections");
        searchFaceInCollection(rekClient, collectionId, sourceImage);
        rekClient.close();
    }

    public static void searchFaceInCollection(RekognitionClient rekClient, String
collectionId, String sourceImage) {
        try {
            InputStream sourceStream = new FileInputStream(new
File(sourceImage));
            SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);
            Image souImage = Image.builder()
                    .bytes(sourceBytes)
                    .build();

            SearchFacesByImageRequest facesByImageRequest =
SearchFacesByImageRequest.builder()
                    .image(souImage)
                    .maxFaces(10)
                    .faceMatchThreshold(70F)
                    .collectionId(collectionId)
                    .build();

            SearchFacesByImageResponse imageResponse =
rekClient.searchFacesByImage(facesByImageRequest);
            System.out.println("Faces matching in the collection");
            List<FaceMatch> faceImageMatches = imageResponse.faceMatches();
            for (FaceMatch face : faceImageMatches) {
                System.out.println("The similarity level is  " +
face.similarity());
                System.out.println();
            }

        } catch (RekognitionException | FileNotFoundException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
```

```
    }
}
```

- Para obtener más información sobre la API, consulta SearchFaceslla Referencia AWS SDK for Java 2.x de la API.

Python

SDK para Python (Boto3)

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
class RekognitionCollection:
    """
    Encapsulates an Amazon Rekognition collection. This class is a thin wrapper
    around parts of the Boto3 Amazon Rekognition API.
    """

    def __init__(self, collection, rekognition_client):
        """
        Initializes a collection object.

        :param collection: Collection data in the format returned by a call to
                           create_collection.
        :param rekognition_client: A Boto3 Rekognition client.
        """
        self.collection_id = collection["CollectionId"]
        self.collection_arn, self.face_count, self.created =
 self._unpack_collection(
            collection
        )
        self.rekognition_client = rekognition_client

    @staticmethod
    def _unpack_collection(collection):
        """
```

```
        Unpacks optional parts of a collection that can be returned by
        describe_collection.

        :param collection: The collection data.
        :return: A tuple of the data in the collection.
        """
        return (
            collection.get("CollectionArn"),
            collection.get("FaceCount", 0),
            collection.get("CreationTimestamp"),
        )


    def search_faces(self, face_id, threshold, max_faces):
        """
        Searches for faces in the collection that match another face from the
        collection.

        :param face_id: The ID of the face in the collection to search for.
        :param threshold: The match confidence must be greater than this value
                          for a face to be included in the results.
        :param max_faces: The maximum number of faces to return.
        :return: The list of matching faces found in the collection. This list
does
                 not contain the face specified by `face_id`.
        """
        try:
            response = self.rekognition_client.search_faces(
                CollectionId=self.collection_id,
                FaceId=face_id,
                FaceMatchThreshold=threshold,
                MaxFaces=max_faces,
            )
            faces = [RekognitionFace(face["Face"]) for face in
response["FaceMatches"]]
            logger.info(
                "Found %s faces in %s that match %s.",
                len(faces),
                self.collection_id,
                face_id,
            )
        except ClientError:
            logger.exception(
                "Couldn't search for faces in %s that match %s.",
```

```
                self.collection_id,
                face_id,
            )
            raise
        else:
            return faces
```

- Para obtener más información sobre la API, consulta SearchFaceslala AWS Referencia de API de SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
   TRY.
       oo_result = lo_rek->searchfaces(
         iv_collectionid = iv_collection_id
         iv_faceid = iv_face_id
         iv_facematchthreshold = iv_threshold
         iv_maxfaces = iv_max_faces ).

       DATA(lt_face_matches) = oo_result->get_facematches( ).
       DATA(lv_match_count2) = lines( lt_face_matches ).
       DATA(lv_msg5) = |Face search completed: { lv_match_count2 } match(es)
found.|.
       MESSAGE lv_msg5 TYPE 'I'.
     CATCH /aws1/cx_rekresourcenotfoundex.
       MESSAGE 'Collection or face not found.' TYPE 'E'.
     CATCH /aws1/cx_rekinvalidparameterex.
       MESSAGE 'Invalid parameter value.' TYPE 'E'.
   ENDTRY.
```

- Para obtener más información sobre la API, consulte SearchFaceslа referencia sobre la API ABAP del AWS SDK para SAP.

Para obtener una lista completa de guías para desarrolladores del AWS SDK y ejemplos de código, consulteUso de Rekognition con un SDK AWS. En este tema también se incluye información sobre cómo comenzar a utilizar el SDK y detalles sobre sus versiones anteriores.

Úselo **`SearchFacesByImage`** con un AWS SDK o CLI

Los siguientes ejemplos de código muestran cómo utilizar `SearchFacesByImage`.

Para obtener más información, consulte Búsqueda de un rostro (imagen).

.NET

SDK para .NET

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
using System;
using System.Threading.Tasks;
using Amazon.Rekognition;
using Amazon.Rekognition.Model;

/// <summary>
/// Uses the Amazon Rekognition Service to search for images matching those
/// in a collection.
/// </summary>
public class SearchFacesMatchingImage
{
    public static async Task Main()
    {
        string collectionId = "MyCollection";
        string bucket = "amzn-s3-demo-bucket";
        string photo = "input.jpg";

        var rekognitionClient = new AmazonRekognitionClient();
```

```
            // Get an image object from S3 bucket.
            var image = new Image()
            {
                S3Object = new S3Object()
                {
                    Bucket = bucket,
                    Name = photo,
                },
            };

            var searchFacesByImageRequest = new SearchFacesByImageRequest()
            {
                CollectionId = collectionId,
                Image = image,
                FaceMatchThreshold = 70F,
                MaxFaces = 2,
            };

            SearchFacesByImageResponse searchFacesByImageResponse = await
rekognitionClient.SearchFacesByImageAsync(searchFacesByImageRequest);

            Console.WriteLine("Faces matching largest face in image from " +
photo);
            searchFacesByImageResponse.FaceMatches.ForEach(face =>
            {
                Console.WriteLine($"FaceId: {face.Face.FaceId}, Similarity:
{face.Similarity}");
            });
        }
    }
```

- Para obtener más información sobre la API, consulta SearchFacesByImagela Referencia AWS SDK para .NET de la API.

CLI

AWS CLI

Búsqueda de rostros en una colección que coincida con el rostro de mayor tamaño en una imagen.

El siguiente comando `search-faces-by-image` busca rostros en una colección que coincidan con el rostro más grande de la imagen especificada:

```
aws rekognition search-faces-by-image \
    --image '{"S3Object":
{"Bucket":"MyImageS3Bucket","Name":"ExamplePerson.jpg"}}' \
    --collection-id MyFaceImageCollection

{
    "SearchedFaceBoundingBox": {
        "Width": 0.18562500178813934,
        "Top": 0.1618015021085739,
        "Left": 0.5575000047683716,
        "Height": 0.24770642817020416
    },
    "SearchedFaceConfidence": 99.993408203125,
    "FaceMatches": [
        {
            "Face": {
                "BoundingBox": {
                    "Width": 0.18562500178813934,
                    "Top": 0.1618019938468933,
                    "Left": 0.5575000047683716,
                    "Height": 0.24770599603652954
                },
                "FaceId": "ce7ed422-2132-4a11-ab14-06c5c410f29f",
                "ExternalImageId": "example-image.jpg",
                "Confidence": 99.99340057373047,
                "ImageId": "8d67061e-90d2-598f-9fbd-29c8497039c0"
            },
            "Similarity": 99.97913360595703
        },
        {
            "Face": {
                "BoundingBox": {
                    "Width": 0.18562500178813934,
                    "Top": 0.1618019938468933,
                    "Left": 0.5575000047683716,
                    "Height": 0.24770599603652954
                },
                "FaceId": "13692fe4-990a-4679-b14a-5ac23d135eab",
                "ExternalImageId": "image3.jpg",
                "Confidence": 99.99340057373047,
```

```
            "ImageId": "8df18239-9ad1-5acd-a46a-6581ff98f51b"
        },
        "Similarity": 99.97913360595703
    },
    {
        "Face": {
            "BoundingBox": {
                "Width": 0.41499999165534973,
                "Top": 0.09187500178813934,
                "Left": 0.28083300590515137,
                "Height": 0.3112500011920929
            },
            "FaceId": "8d3cfc70-4ba8-4b36-9644-90fba29c2dac",
            "ExternalImageId": "image2.jpg",
            "Confidence": 99.99769592285156,
            "ImageId": "a294da46-2cb1-5cc4-9045-61d7ca567662"
        },
        "Similarity": 99.18069458007812
    },
    {
        "Face": {
            "BoundingBox": {
                "Width": 0.48166701197624207,
                "Top": 0.20999999344348907,
                "Left": 0.21250000596046448,
                "Height": 0.36125001311302185
            },
            "FaceId": "bd4ceb4d-9acc-4ab7-8ef8-1c2d2ba0a66a",
            "ExternalImageId": "image1.jpg",
            "Confidence": 99.99949645996094,
            "ImageId": "5e1a7588-e5a0-5ee3-bd00-c642518dfe3a"
        },
        "Similarity": 98.66607666015625
    },
    {
        "Face": {
            "BoundingBox": {
                "Width": 0.5349419713020325,
                "Top": 0.29124999046325684,
                "Left": 0.16389399766921997,
                "Height": 0.40187498927116394
            },
            "FaceId": "745f7509-b1fa-44e0-8b95-367b1359638a",
            "ExternalImageId": "image9.jpg",
```

```
            "Confidence": 99.99979400634766,
            "ImageId": "67a34327-48d1-5179-b042-01e52ccfeada"
        },
        "Similarity": 98.24278259277344
    },
    {
        "Face": {
            "BoundingBox": {
                "Width": 0.5307819843292236,
                "Top": 0.2862499952316284,
                "Left": 0.1564060002565384,
                "Height": 0.3987500071525574
            },
            "FaceId": "2eb5f3fd-e2a9-4b1c-a89f-afa0a518fe06",
            "ExternalImageId": "image10.jpg",
            "Confidence": 99.99970245361328,
            "ImageId": "3c314792-197d-528d-bbb6-798ed012c150"
        },
        "Similarity": 98.10665893554688
    },
    {
        "Face": {
            "BoundingBox": {
                "Width": 0.5074880123138428,
                "Top": 0.3774999976158142,
                "Left": 0.18302799761295319,
                "Height": 0.3812499940395355
            },
            "FaceId": "086261e8-6deb-4bc0-ac73-ab22323cc38d",
            "ExternalImageId": "image6.jpg",
            "Confidence": 99.99930572509766,
            "ImageId": "ae1593b0-a8f6-5e24-a306-abf529e276fa"
        },
        "Similarity": 98.10526275634766
    },
    {
        "Face": {
            "BoundingBox": {
                "Width": 0.5574039816856384,
                "Top": 0.37187498807907104,
                "Left": 0.14559100568294525,
                "Height": 0.4181250035762787
            },
            "FaceId": "11c4bd3c-19c5-4eb8-aecc-24feb93a26e1",
```

```
                "ExternalImageId": "image5.jpg",
                "Confidence": 99.99960327148438,
                "ImageId": "80739b4d-883f-5b78-97cf-5124038e26b9"
            },
            "Similarity": 97.94659423828125
        },
        {
            "Face": {
                "BoundingBox": {
                    "Width": 0.5773710012435913,
                    "Top": 0.34437501430511475,
                    "Left": 0.12396000325679779,
                    "Height": 0.4337500035762787
                },
                "FaceId": "57189455-42b0-4839-a86c-abda48b13174",
                "ExternalImageId": "image8.jpg",
                "Confidence": 100.0,
                "ImageId": "0aff2f37-e7a2-5dbc-a3a3-4ef6ec18eaa0"
            },
            "Similarity": 97.93476867675781
        }
    ],

 "FaceModelVersion": "3.0"
}
```

Para obtener más información, consulte Búsqueda de un rostro utilizando una imagen en la Guía para desarrolladores de Amazon Rekognition.

- Para obtener más información sobre la API, consulta SearchFacesByImagela Referencia de AWS CLI comandos.

Java

SDK para Java 2.x

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.SearchFacesRequest;
import software.amazon.awssdk.services.rekognition.model.SearchFacesResponse;
import software.amazon.awssdk.services.rekognition.model.FaceMatch;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class SearchFaceMatchingIdCollection {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <collectionId> <sourceImage>

                Where:
                   collectionId - The id of the collection. \s
                   sourceImage - The path to the image (for example, C:\\AWS\
\pic1.png).\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String collectionId = args[0];
        String faceId = args[1];
        Region region = Region.US_WEST_2;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        System.out.println("Searching for a face in a collections");
        searchFacebyId(rekClient, collectionId, faceId);
```

```
        rekClient.close();
    }

    public static void searchFacebyId(RekognitionClient rekClient, String
 collectionId, String faceId) {
        try {
            SearchFacesRequest searchFacesRequest = SearchFacesRequest.builder()
                    .collectionId(collectionId)
                    .faceId(faceId)
                    .faceMatchThreshold(70F)
                    .maxFaces(2)
                    .build();

            SearchFacesResponse imageResponse =
 rekClient.searchFaces(searchFacesRequest);
            System.out.println("Faces matching in the collection");
            List<FaceMatch> faceImageMatches = imageResponse.faceMatches();
            for (FaceMatch face : faceImageMatches) {
                System.out.println("The similarity level is  " +
 face.similarity());
                System.out.println();
            }

        } catch (RekognitionException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Para obtener más información sobre la API, consulta SearchFacesByImagela Referencia AWS SDK for Java 2.x de la API.

Python

SDK para Python (Boto3)

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
class RekognitionCollection:
    """
    Encapsulates an Amazon Rekognition collection. This class is a thin wrapper
    around parts of the Boto3 Amazon Rekognition API.
    """

    def __init__(self, collection, rekognition_client):
        """
        Initializes a collection object.

        :param collection: Collection data in the format returned by a call to
                           create_collection.
        :param rekognition_client: A Boto3 Rekognition client.
        """
        self.collection_id = collection["CollectionId"]
        self.collection_arn, self.face_count, self.created =
 self._unpack_collection(
            collection
        )
        self.rekognition_client = rekognition_client

    @staticmethod
    def _unpack_collection(collection):
        """
        Unpacks optional parts of a collection that can be returned by
        describe_collection.

        :param collection: The collection data.
        :return: A tuple of the data in the collection.
        """
        return (
            collection.get("CollectionArn"),
```

```
            collection.get("FaceCount", 0),
            collection.get("CreationTimestamp"),
        )


    def search_faces_by_image(self, image, threshold, max_faces):
        """
        Searches for faces in the collection that match the largest face in the
        reference image.

        :param image: The image that contains the reference face to search for.
        :param threshold: The match confidence must be greater than this value
                          for a face to be included in the results.
        :param max_faces: The maximum number of faces to return.
        :return: A tuple. The first element is the face found in the reference
image.
                 The second element is the list of matching faces found in the
                 collection.
        """
        try:
            response = self.rekognition_client.search_faces_by_image(
                CollectionId=self.collection_id,
                Image=image.image,
                FaceMatchThreshold=threshold,
                MaxFaces=max_faces,
            )
            image_face = RekognitionFace(
                {
                    "BoundingBox": response["SearchedFaceBoundingBox"],
                    "Confidence": response["SearchedFaceConfidence"],
                }
            )
            collection_faces = [
                RekognitionFace(face["Face"]) for face in response["FaceMatches"]
            ]
            logger.info(
                "Found %s faces in the collection that match the largest "
                "face in %s.",
                len(collection_faces),
                image.image_name,
            )
        except ClientError:
            logger.exception(
                "Couldn't search for faces in %s that match %s.",
```

```
                self.collection_id,
                image.image_name,
            )
            raise
        else:
            return image_face, collection_faces
```

- Para obtener más información sobre la API, consulta SearchFacesByImagela AWS Referencia de API de SDK for Python (Boto3).

SAP ABAP

SDK para SAP ABAP

> **Note**
>
> Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
TRY.
    " Create S3 object reference for the image
    DATA(lo_s3object) = NEW /aws1/cl_reks3object(
      iv_bucket = iv_s3_bucket
      iv_name = iv_s3_key ).

    " Create image object
    DATA(lo_image) = NEW /aws1/cl_rekimage(
      io_s3object = lo_s3object ).

    " Search for matching faces
    oo_result = lo_rek->searchfacesbyimage(
      iv_collectionid = iv_collection_id
      io_image = lo_image
      iv_facematchthreshold = iv_threshold
      iv_maxfaces = iv_max_faces ).

    DATA(lt_face_matches) = oo_result->get_facematches( ).
    DATA(lv_match_count) = lines( lt_face_matches ).
```

```
        DATA(lv_msg4) = |Face search completed: { lv_match_count } match(es)
found.|.
        MESSAGE lv_msg4 TYPE 'I'.
      CATCH /aws1/cx_rekresourcenotfoundex.
        MESSAGE 'Collection not found.' TYPE 'E'.
      CATCH /aws1/cx_rekinvalids3objectex.
        MESSAGE 'Invalid S3 object.' TYPE 'E'.
      CATCH /aws1/cx_rekinvalidparameterex.
        MESSAGE 'Invalid parameter value.' TYPE 'E'.
    ENDTRY.
```

- Para obtener más información sobre la API, consulte SearchFacesByImagela referencia sobre la API ABAP del AWS SDK para SAP.

Para obtener una lista completa de guías para desarrolladores del AWS SDK y ejemplos de código, consulteUso de Rekognition con un SDK AWS. En este tema también se incluye información sobre cómo comenzar a utilizar el SDK y detalles sobre sus versiones anteriores.

Escenarios para el uso de Amazon Rekognition AWS SDKs

Los siguientes ejemplos de código muestran cómo implementar escenarios comunes en Amazon Rekognition con. AWS SDKs Estos escenarios muestran cómo llevar a cabo tareas específicas con llamadas a varias funciones dentro de Amazon Rekognition o en combinación con otros Servicios de AWS. En cada escenario se incluye un enlace al código fuente completo, con instrucciones de configuración y ejecución del código.

Los escenarios requieren un nivel intermedio de experiencia para entender las acciones de servicio en su contexto.

Ejemplos

- Crea una colección de Amazon Rekognition y encuentra rostros en ella con un SDK AWS
- Creación de una aplicación de administración de activos fotográficos que permita a los usuarios administrar las fotos mediante etiquetas
- Detecte el PPE en las imágenes con Amazon Rekognition AWS mediante un SDK
- Detecte y muestre elementos en imágenes con Amazon Rekognition AWS mediante un SDK
- Detecta rostros en una imagen mediante un AWS SDK
- Detecte información en vídeos con Amazon Rekognition y el SDK AWS

- Detecte objetos en imágenes con Amazon Rekognition AWS mediante un SDK
- Detecte personas y objetos en un vídeo con Amazon Rekognition AWS mediante un SDK
- Guarde el EXIF y otra información de imagen mediante un SDK AWS

Crea una colección de Amazon Rekognition y encuentra rostros en ella con un SDK AWS

En el siguiente ejemplo de código, se muestra cómo:

- Crear una colección de Amazon Rekognition.
- Añadir imágenes a la colección y detectar rostros en ella.
- Buscar rostros en la colección que coincidan con una imagen de referencia.
- Eliminar una colección.

Para obtener más información, consulte Buscar rostros en una colección.

Python

SDK para Python (Boto3)

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

Cree clases que incluyan las funciones de Amazon Rekognition.

```
import logging
from pprint import pprint
import boto3
from botocore.exceptions import ClientError
from rekognition_objects import RekognitionFace
from rekognition_image_detection import RekognitionImage

logger = logging.getLogger(__name__)
```

```
class RekognitionImage:
    """
    Encapsulates an Amazon Rekognition image. This class is a thin wrapper
    around parts of the Boto3 Amazon Rekognition API.
    """

    def __init__(self, image, image_name, rekognition_client):
        """
        Initializes the image object.

        :param image: Data that defines the image, either the image bytes or
                      an Amazon S3 bucket and object key.
        :param image_name: The name of the image.
        :param rekognition_client: A Boto3 Rekognition client.
        """
        self.image = image
        self.image_name = image_name
        self.rekognition_client = rekognition_client


    @classmethod
    def from_file(cls, image_file_name, rekognition_client, image_name=None):
        """
        Creates a RekognitionImage object from a local file.

        :param image_file_name: The file name of the image. The file is opened
 and its
                                bytes are read.
        :param rekognition_client: A Boto3 Rekognition client.
        :param image_name: The name of the image. If this is not specified, the
                           file name is used as the image name.
        :return: The RekognitionImage object, initialized with image bytes from
 the
                 file.
        """
        with open(image_file_name, "rb") as img_file:
            image = {"Bytes": img_file.read()}
        name = image_file_name if image_name is None else image_name
        return cls(image, name, rekognition_client)


class RekognitionCollectionManager:
    """
    Encapsulates Amazon Rekognition collection management functions.
```

```
    This class is a thin wrapper around parts of the Boto3 Amazon Rekognition
API.
    """

    def __init__(self, rekognition_client):
        """
        Initializes the collection manager object.

        :param rekognition_client: A Boto3 Rekognition client.
        """
        self.rekognition_client = rekognition_client


    def create_collection(self, collection_id):
        """
        Creates an empty collection.

        :param collection_id: Text that identifies the collection.
        :return: The newly created collection.
        """
        try:
            response = self.rekognition_client.create_collection(
                CollectionId=collection_id
            )
            response["CollectionId"] = collection_id
            collection = RekognitionCollection(response, self.rekognition_client)
            logger.info("Created collection %s.", collection_id)
        except ClientError:
            logger.exception("Couldn't create collection %s.", collection_id)
            raise
        else:
            return collection


    def list_collections(self, max_results):
        """
        Lists collections for the current account.

        :param max_results: The maximum number of collections to return.
        :return: The list of collections for the current account.
        """
        try:
            response =
self.rekognition_client.list_collections(MaxResults=max_results)
```

```
            collections = [
                RekognitionCollection({"CollectionId": col_id},
 self.rekognition_client)
                for col_id in response["CollectionIds"]
            ]
        except ClientError:
            logger.exception("Couldn't list collections.")
            raise
        else:
            return collections



class RekognitionCollection:
    """
    Encapsulates an Amazon Rekognition collection. This class is a thin wrapper
    around parts of the Boto3 Amazon Rekognition API.
    """

    def __init__(self, collection, rekognition_client):
        """
        Initializes a collection object.

        :param collection: Collection data in the format returned by a call to
                           create_collection.
        :param rekognition_client: A Boto3 Rekognition client.
        """
        self.collection_id = collection["CollectionId"]
        self.collection_arn, self.face_count, self.created =
 self._unpack_collection(
            collection
        )
        self.rekognition_client = rekognition_client

    @staticmethod
    def _unpack_collection(collection):
        """
        Unpacks optional parts of a collection that can be returned by
        describe_collection.

        :param collection: The collection data.
        :return: A tuple of the data in the collection.
        """
        return (
```

```
            collection.get("CollectionArn"),
            collection.get("FaceCount", 0),
            collection.get("CreationTimestamp"),
        )


    def to_dict(self):
        """
        Renders parts of the collection data to a dict.

        :return: The collection data as a dict.
        """
        rendering = {
            "collection_id": self.collection_id,
            "collection_arn": self.collection_arn,
            "face_count": self.face_count,
            "created": self.created,
        }
        return rendering


    def describe_collection(self):
        """
        Gets data about the collection from the Amazon Rekognition service.

        :return: The collection rendered as a dict.
        """
        try:
            response = self.rekognition_client.describe_collection(
                CollectionId=self.collection_id
            )
            # Work around capitalization of Arn vs. ARN
            response["CollectionArn"] = response.get("CollectionARN")
            (
                self.collection_arn,
                self.face_count,
                self.created,
            ) = self._unpack_collection(response)
            logger.info("Got data for collection %s.", self.collection_id)
        except ClientError:
            logger.exception("Couldn't get data for collection %s.",
self.collection_id)
            raise
        else:
```

```
            return self.to_dict()


    def delete_collection(self):
        """
        Deletes the collection.
        """
        try:

self.rekognition_client.delete_collection(CollectionId=self.collection_id)
            logger.info("Deleted collection %s.", self.collection_id)
            self.collection_id = None
        except ClientError:
            logger.exception("Couldn't delete collection %s.",
self.collection_id)
            raise


    def index_faces(self, image, max_faces):
        """
        Finds faces in the specified image, indexes them, and stores them in the
        collection.

        :param image: The image to index.
        :param max_faces: The maximum number of faces to index.
        :return: A tuple. The first element is a list of indexed faces.
                 The second element is a list of faces that couldn't be indexed.
        """
        try:
            response = self.rekognition_client.index_faces(
                CollectionId=self.collection_id,
                Image=image.image,
                ExternalImageId=image.image_name,
                MaxFaces=max_faces,
                DetectionAttributes=["ALL"],
            )
            indexed_faces = [
                RekognitionFace({**face["Face"], **face["FaceDetail"]})
                for face in response["FaceRecords"]
            ]
            unindexed_faces = [
                RekognitionFace(face["FaceDetail"])
                for face in response["UnindexedFaces"]
            ]
```

```
            logger.info(
                "Indexed %s faces in %s. Could not index %s faces.",
                len(indexed_faces),
                image.image_name,
                len(unindexed_faces),
            )
        except ClientError:
            logger.exception("Couldn't index faces in image %s.",
image.image_name)
            raise
        else:
            return indexed_faces, unindexed_faces


    def list_faces(self, max_results):
        """
        Lists the faces currently indexed in the collection.

        :param max_results: The maximum number of faces to return.
        :return: The list of faces in the collection.
        """
        try:
            response = self.rekognition_client.list_faces(
                CollectionId=self.collection_id, MaxResults=max_results
            )
            faces = [RekognitionFace(face) for face in response["Faces"]]
            logger.info(
                "Found %s faces in collection %s.", len(faces),
self.collection_id
            )
        except ClientError:
            logger.exception(
                "Couldn't list faces in collection %s.", self.collection_id
            )
            raise
        else:
            return faces


    def search_faces(self, face_id, threshold, max_faces):
        """
        Searches for faces in the collection that match another face from the
        collection.
```

```
        :param face_id: The ID of the face in the collection to search for.
        :param threshold: The match confidence must be greater than this value
                          for a face to be included in the results.
        :param max_faces: The maximum number of faces to return.
        :return: The list of matching faces found in the collection. This list
does
                 not contain the face specified by `face_id`.
        """
        try:
            response = self.rekognition_client.search_faces(
                CollectionId=self.collection_id,
                FaceId=face_id,
                FaceMatchThreshold=threshold,
                MaxFaces=max_faces,
            )
            faces = [RekognitionFace(face["Face"]) for face in
response["FaceMatches"]]
            logger.info(
                "Found %s faces in %s that match %s.",
                len(faces),
                self.collection_id,
                face_id,
            )
        except ClientError:
            logger.exception(
                "Couldn't search for faces in %s that match %s.",
                self.collection_id,
                face_id,
            )
            raise
        else:
            return faces


    def search_faces_by_image(self, image, threshold, max_faces):
        """
        Searches for faces in the collection that match the largest face in the
        reference image.

        :param image: The image that contains the reference face to search for.
        :param threshold: The match confidence must be greater than this value
                          for a face to be included in the results.
        :param max_faces: The maximum number of faces to return.
```

```
        :return: A tuple. The first element is the face found in the reference
 image.
                 The second element is the list of matching faces found in the
                 collection.
        """
        try:
            response = self.rekognition_client.search_faces_by_image(
                CollectionId=self.collection_id,
                Image=image.image,
                FaceMatchThreshold=threshold,
                MaxFaces=max_faces,
            )
            image_face = RekognitionFace(
                {
                    "BoundingBox": response["SearchedFaceBoundingBox"],
                    "Confidence": response["SearchedFaceConfidence"],
                }
            )
            collection_faces = [
                RekognitionFace(face["Face"]) for face in response["FaceMatches"]
            ]
            logger.info(
                "Found %s faces in the collection that match the largest "
                "face in %s.",
                len(collection_faces),
                image.image_name,
            )
        except ClientError:
            logger.exception(
                "Couldn't search for faces in %s that match %s.",
                self.collection_id,
                image.image_name,
            )
            raise
        else:
            return image_face, collection_faces


class RekognitionFace:
    """Encapsulates an Amazon Rekognition face."""

    def __init__(self, face, timestamp=None):
        """
        Initializes the face object.
```

```
        :param face: Face data, in the format returned by Amazon Rekognition
                     functions.
        :param timestamp: The time when the face was detected, if the face was
                          detected in a video.
        """
        self.bounding_box = face.get("BoundingBox")
        self.confidence = face.get("Confidence")
        self.landmarks = face.get("Landmarks")
        self.pose = face.get("Pose")
        self.quality = face.get("Quality")
        age_range = face.get("AgeRange")
        if age_range is not None:
            self.age_range = (age_range.get("Low"), age_range.get("High"))
        else:
            self.age_range = None
        self.smile = face.get("Smile", {}).get("Value")
        self.eyeglasses = face.get("Eyeglasses", {}).get("Value")
        self.sunglasses = face.get("Sunglasses", {}).get("Value")
        self.gender = face.get("Gender", {}).get("Value", None)
        self.beard = face.get("Beard", {}).get("Value")
        self.mustache = face.get("Mustache", {}).get("Value")
        self.eyes_open = face.get("EyesOpen", {}).get("Value")
        self.mouth_open = face.get("MouthOpen", {}).get("Value")
        self.emotions = [
            emo.get("Type")
            for emo in face.get("Emotions", [])
            if emo.get("Confidence", 0) > 50
        ]
        self.face_id = face.get("FaceId")
        self.image_id = face.get("ImageId")
        self.timestamp = timestamp

    def to_dict(self):
        """
        Renders some of the face data to a dict.

        :return: A dict that contains the face data.
        """
        rendering = {}
        if self.bounding_box is not None:
            rendering["bounding_box"] = self.bounding_box
        if self.age_range is not None:
            rendering["age"] = f"{self.age_range[0]} - {self.age_range[1]}"
```

```
        if self.gender is not None:
            rendering["gender"] = self.gender
        if self.emotions:
            rendering["emotions"] = self.emotions
        if self.face_id is not None:
            rendering["face_id"] = self.face_id
        if self.image_id is not None:
            rendering["image_id"] = self.image_id
        if self.timestamp is not None:
            rendering["timestamp"] = self.timestamp
        has = []
        if self.smile:
            has.append("smile")
        if self.eyeglasses:
            has.append("eyeglasses")
        if self.sunglasses:
            has.append("sunglasses")
        if self.beard:
            has.append("beard")
        if self.mustache:
            has.append("mustache")
        if self.eyes_open:
            has.append("open eyes")
        if self.mouth_open:
            has.append("open mouth")
        if has:
            rendering["has"] = has
        return rendering
```

Utilice las clases de contenedor para crear una colección de rostros a partir de un conjunto de imágenes y, a continuación, busque rostros en la colección.

```
def usage_demo():
    print("-" * 88)
    print("Welcome to the Amazon Rekognition face collection demo!")
    print("-" * 88)

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

    rekognition_client = boto3.client("rekognition")
```

```
    images = [
        RekognitionImage.from_file(
            ".media/pexels-agung-pandit-wiguna-1128316.jpg",
            rekognition_client,
            image_name="sitting",
        ),
        RekognitionImage.from_file(
            ".media/pexels-agung-pandit-wiguna-1128317.jpg",
            rekognition_client,
            image_name="hopping",
        ),
        RekognitionImage.from_file(
            ".media/pexels-agung-pandit-wiguna-1128318.jpg",
            rekognition_client,
            image_name="biking",
        ),
    ]

    collection_mgr = RekognitionCollectionManager(rekognition_client)
    collection = collection_mgr.create_collection("doc-example-collection-demo")
    print(f"Created collection {collection.collection_id}:")
    pprint(collection.describe_collection())

    print("Indexing faces from three images:")
    for image in images:
        collection.index_faces(image, 10)
    print("Listing faces in collection:")
    faces = collection.list_faces(10)
    for face in faces:
        pprint(face.to_dict())
    input("Press Enter to continue.")

    print(
        f"Searching for faces in the collection that match the first face in the
"
        f"list (Face ID: {faces[0].face_id}."
    )
    found_faces = collection.search_faces(faces[0].face_id, 80, 10)
    print(f"Found {len(found_faces)} matching faces.")
    for face in found_faces:
        pprint(face.to_dict())
    input("Press Enter to continue.")

    print(
```

```
        f"Searching for faces in the collection that match the largest face in "
        f"{images[0].image_name}."
    )
    image_face, match_faces = collection.search_faces_by_image(images[0], 80, 10)
    print(f"The largest face in {images[0].image_name} is:")
    pprint(image_face.to_dict())
    print(f"Found {len(match_faces)} matching faces.")
    for face in match_faces:
        pprint(face.to_dict())
    input("Press Enter to continue.")

    collection.delete_collection()
    print("Thanks for watching!")
    print("-" * 88)
```

Para obtener una lista completa de guías para desarrolladores del AWS SDK y ejemplos de código, consulteUso de Rekognition con un SDK AWS. En este tema también se incluye información sobre cómo comenzar a utilizar el SDK y detalles sobre sus versiones anteriores.

Creación de una aplicación de administración de activos fotográficos que permita a los usuarios administrar las fotos mediante etiquetas

En los siguientes ejemplos de código se muestra cómo crear una aplicación sin servidor que permita a los usuarios administrar fotos mediante etiquetas.

.NET

SDK para .NET

Muestra cómo desarrollar una aplicación de administración de activos fotográficos que detecte las etiquetas de las imágenes mediante Amazon Rekognition y las almacene para su posterior recuperación.

Para ver el código fuente completo y las instrucciones sobre cómo configurarlo y ejecutarlo, consulta el ejemplo completo en GitHub.

Para profundizar en el origen de este ejemplo, consulte la publicación en Comunidad de AWS.

Servicios utilizados en este ejemplo

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

C++

SDK para C++

Muestra cómo desarrollar una aplicación de administración de activos fotográficos que detecte las etiquetas de las imágenes mediante Amazon Rekognition y las almacene para su posterior recuperación.

Para ver el código fuente completo y las instrucciones sobre cómo configurarlo y ejecutarlo, consulta el ejemplo completo en GitHub.

Para profundizar en el origen de este ejemplo, consulte la publicación en Comunidad de AWS.

Servicios utilizados en este ejemplo

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Java

SDK para Java 2.x

Muestra cómo desarrollar una aplicación de administración de activos fotográficos que detecte las etiquetas de las imágenes mediante Amazon Rekognition y las almacene para su posterior recuperación.

Para ver el código fuente completo y las instrucciones sobre cómo configurarlo y ejecutarlo, consulta el ejemplo completo en GitHub.

Para profundizar en el origen de este ejemplo, consulte la publicación en Comunidad de AWS.

Servicios utilizados en este ejemplo

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

JavaScript

SDK para JavaScript (v3)

Muestra cómo desarrollar una aplicación de administración de activos fotográficos que detecte las etiquetas de las imágenes mediante Amazon Rekognition y las almacene para su posterior recuperación.

Para ver el código fuente completo y las instrucciones sobre cómo configurarlo y ejecutarlo, consulta el ejemplo completo en GitHub.

Para profundizar en el origen de este ejemplo, consulte la publicación en Comunidad de AWS.

Servicios utilizados en este ejemplo

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Kotlin

SDK para Kotlin

Muestra cómo desarrollar una aplicación de administración de activos fotográficos que detecte las etiquetas de las imágenes mediante Amazon Rekognition y las almacene para su posterior recuperación.

Para ver el código fuente completo y las instrucciones sobre cómo configurarlo y ejecutarlo, consulta el ejemplo completo en GitHub.

Para profundizar en el origen de este ejemplo, consulte la publicación en Comunidad de AWS.

Servicios utilizados en este ejemplo

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

PHP

SDK para PHP

Muestra cómo desarrollar una aplicación de administración de activos fotográficos que detecte las etiquetas de las imágenes mediante Amazon Rekognition y las almacene para su posterior recuperación.

Para ver el código fuente completo y las instrucciones sobre cómo configurarlo y ejecutarlo, consulta el ejemplo completo en GitHub.

Para profundizar en el origen de este ejemplo, consulte la publicación en Comunidad de AWS.

Servicios utilizados en este ejemplo

- API Gateway
- DynamoDB

- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Rust

SDK para Rust

Muestra cómo desarrollar una aplicación de administración de activos fotográficos que detecte las etiquetas de las imágenes mediante Amazon Rekognition y las almacene para su posterior recuperación.

Para ver el código fuente completo y las instrucciones sobre cómo configurarlo y ejecutarlo, consulta el ejemplo completo en GitHub.

Para profundizar en el origen de este ejemplo, consulte la publicación en Comunidad de AWS.

Servicios utilizados en este ejemplo

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Para obtener una lista completa de guías para desarrolladores del AWS SDK y ejemplos de código, consulteUso de Rekognition con un SDK AWS. En este tema también se incluye información sobre cómo comenzar a utilizar el SDK y detalles sobre sus versiones anteriores.

Detecte el PPE en las imágenes con Amazon Rekognition AWS mediante un SDK

El siguiente ejemplo de código muestra cómo crear una aplicación que utiliza Amazon Rekognition para detectar equipos de protección individual (EPI) en imágenes.

Java

SDK para Java 2.x

Muestra cómo crear una AWS Lambda función que detecte imágenes con un equipo de protección individual.

Para obtener el código fuente completo y las instrucciones sobre cómo configurarlo y ejecutarlo, consulte el ejemplo completo en GitHub.

Servicios utilizados en este ejemplo

- DynamoDB
- Amazon Rekognition
- Amazon S3
- Amazon SES

Para obtener una lista completa de guías para desarrolladores del AWS SDK y ejemplos de código, consulteUso de Rekognition con un SDK AWS. En este tema también se incluye información sobre cómo comenzar a utilizar el SDK y detalles sobre sus versiones anteriores.

Detecte y muestre elementos en imágenes con Amazon Rekognition AWS mediante un SDK

En el siguiente ejemplo de código, se muestra cómo:

- Detectar elementos en imágenes con Amazon Rekognition.
- Mostrar imágenes y dibujar cuadros delimitadores alrededor de los elementos detectados.

Para obtener más información, consulte Mostrar de cuadros delimitadores.

Python

SDK para Python (Boto3)

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

Cree clases que incluyan las funciones de Amazon Rekognition.

```
import logging
from pprint import pprint
import boto3
from botocore.exceptions import ClientError
import requests

from rekognition_objects import (
    RekognitionFace,
    RekognitionCelebrity,
    RekognitionLabel,
    RekognitionModerationLabel,
    RekognitionText,
    show_bounding_boxes,
    show_polygons,
)

logger = logging.getLogger(__name__)


class RekognitionImage:
    """
    Encapsulates an Amazon Rekognition image. This class is a thin wrapper
    around parts of the Boto3 Amazon Rekognition API.
    """

    def __init__(self, image, image_name, rekognition_client):
        """
        Initializes the image object.

        :param image: Data that defines the image, either the image bytes or
                      an Amazon S3 bucket and object key.
        :param image_name: The name of the image.
        :param rekognition_client: A Boto3 Rekognition client.
        """
        self.image = image
        self.image_name = image_name
        self.rekognition_client = rekognition_client


    @classmethod
    def from_file(cls, image_file_name, rekognition_client, image_name=None):
        """
```

```
        Creates a RekognitionImage object from a local file.

        :param image_file_name: The file name of the image. The file is opened
and its
                                bytes are read.
        :param rekognition_client: A Boto3 Rekognition client.
        :param image_name: The name of the image. If this is not specified, the
                           file name is used as the image name.
        :return: The RekognitionImage object, initialized with image bytes from
the
                 file.
        """
        with open(image_file_name, "rb") as img_file:
            image = {"Bytes": img_file.read()}
        name = image_file_name if image_name is None else image_name
        return cls(image, name, rekognition_client)


    @classmethod
    def from_bucket(cls, s3_object, rekognition_client):
        """
        Creates a RekognitionImage object from an Amazon S3 object.

        :param s3_object: An Amazon S3 object that identifies the image. The
image
                          is not retrieved until needed for a later call.
        :param rekognition_client: A Boto3 Rekognition client.
        :return: The RekognitionImage object, initialized with Amazon S3 object
data.
        """
        image = {"S3Object": {"Bucket": s3_object.bucket_name, "Name":
s3_object.key}}
        return cls(image, s3_object.key, rekognition_client)


    def detect_faces(self):
        """
        Detects faces in the image.

        :return: The list of faces found in the image.
        """
        try:
            response = self.rekognition_client.detect_faces(
                Image=self.image, Attributes=["ALL"]
```

```
            )
            faces = [RekognitionFace(face) for face in response["FaceDetails"]]
            logger.info("Detected %s faces.", len(faces))
        except ClientError:
            logger.exception("Couldn't detect faces in %s.", self.image_name)
            raise
        else:
            return faces


    def detect_labels(self, max_labels):
        """
        Detects labels in the image. Labels are objects and people.

        :param max_labels: The maximum number of labels to return.
        :return: The list of labels detected in the image.
        """
        try:
            response = self.rekognition_client.detect_labels(
                Image=self.image, MaxLabels=max_labels
            )
            labels = [RekognitionLabel(label) for label in response["Labels"]]
            logger.info("Found %s labels in %s.", len(labels), self.image_name)
        except ClientError:
            logger.info("Couldn't detect labels in %s.", self.image_name)
            raise
        else:
            return labels


    def recognize_celebrities(self):
        """
        Detects celebrities in the image.

        :return: A tuple. The first element is the list of celebrities found in
                 the image. The second element is the list of faces that were
                 detected but did not match any known celebrities.
        """
        try:
            response =
self.rekognition_client.recognize_celebrities(Image=self.image)
            celebrities = [
                RekognitionCelebrity(celeb) for celeb in
response["CelebrityFaces"]
```

```
            ]
            other_faces = [
                RekognitionFace(face) for face in response["UnrecognizedFaces"]
            ]
            logger.info(
                "Found %s celebrities and %s other faces in %s.",
                len(celebrities),
                len(other_faces),
                self.image_name,
            )
        except ClientError:
            logger.exception("Couldn't detect celebrities in %s.",
self.image_name)
            raise
        else:
            return celebrities, other_faces



    def compare_faces(self, target_image, similarity):
        """
        Compares faces in the image with the largest face in the target image.

        :param target_image: The target image to compare against.
        :param similarity: Faces in the image must have a similarity value
greater
                           than this value to be included in the results.
        :return: A tuple. The first element is the list of faces that match the
                 reference image. The second element is the list of faces that
have
                 a similarity value below the specified threshold.
        """
        try:
            response = self.rekognition_client.compare_faces(
                SourceImage=self.image,
                TargetImage=target_image.image,
                SimilarityThreshold=similarity,
            )
            matches = [
                RekognitionFace(match["Face"]) for match in
response["FaceMatches"]
            ]
            unmatches = [RekognitionFace(face) for face in
response["UnmatchedFaces"]]
```

```
            logger.info(
                "Found %s matched faces and %s unmatched faces.",
                len(matches),
                len(unmatches),
            )
        except ClientError:
            logger.exception(
                "Couldn't match faces from %s to %s.",
                self.image_name,
                target_image.image_name,
            )
            raise
        else:
            return matches, unmatches


    def detect_moderation_labels(self):
        """
        Detects moderation labels in the image. Moderation labels identify
content
        that may be inappropriate for some audiences.

        :return: The list of moderation labels found in the image.
        """
        try:
            response = self.rekognition_client.detect_moderation_labels(
                Image=self.image
            )
            labels = [
                RekognitionModerationLabel(label)
                for label in response["ModerationLabels"]
            ]
            logger.info(
                "Found %s moderation labels in %s.", len(labels), self.image_name
            )
        except ClientError:
            logger.exception(
                "Couldn't detect moderation labels in %s.", self.image_name
            )
            raise
        else:
            return labels
```

```
    def detect_text(self):
        """
        Detects text in the image.

        :return The list of text elements found in the image.
        """
        try:
            response = self.rekognition_client.detect_text(Image=self.image)
            texts = [RekognitionText(text) for text in
 response["TextDetections"]]
            logger.info("Found %s texts in %s.", len(texts), self.image_name)
        except ClientError:
            logger.exception("Couldn't detect text in %s.", self.image_name)
            raise
        else:
            return texts
```

Cree funciones auxiliares para dibujar cuadros delimitadores y polígonos.

```
import io
import logging
from PIL import Image, ImageDraw

logger = logging.getLogger(__name__)


def show_bounding_boxes(image_bytes, box_sets, colors):
    """
    Draws bounding boxes on an image and shows it with the default image viewer.

    :param image_bytes: The image to draw, as bytes.
    :param box_sets: A list of lists of bounding boxes to draw on the image.
    :param colors: A list of colors to use to draw the bounding boxes.
    """
    image = Image.open(io.BytesIO(image_bytes))
    draw = ImageDraw.Draw(image)
    for boxes, color in zip(box_sets, colors):
        for box in boxes:
            left = image.width * box["Left"]
            top = image.height * box["Top"]
            right = (image.width * box["Width"]) + left
```

```
            bottom = (image.height * box["Height"]) + top
            draw.rectangle([left, top, right, bottom], outline=color, width=3)
    image.show()



def show_polygons(image_bytes, polygons, color):
    """
    Draws polygons on an image and shows it with the default image viewer.

    :param image_bytes: The image to draw, as bytes.
    :param polygons: The list of polygons to draw on the image.
    :param color: The color to use to draw the polygons.
    """
    image = Image.open(io.BytesIO(image_bytes))
    draw = ImageDraw.Draw(image)
    for polygon in polygons:
        draw.polygon(
            [
                (image.width * point["X"], image.height * point["Y"])
                for point in polygon
            ],
            outline=color,
        )
    image.show()
```

Cree clases para analizar los objetos devueltos por Amazon Rekognition.

```
class RekognitionFace:
    """Encapsulates an Amazon Rekognition face."""

    def __init__(self, face, timestamp=None):
        """
        Initializes the face object.

        :param face: Face data, in the format returned by Amazon Rekognition
                     functions.
        :param timestamp: The time when the face was detected, if the face was
                          detected in a video.
        """
```

```
        self.bounding_box = face.get("BoundingBox")
        self.confidence = face.get("Confidence")
        self.landmarks = face.get("Landmarks")
        self.pose = face.get("Pose")
        self.quality = face.get("Quality")
        age_range = face.get("AgeRange")
        if age_range is not None:
            self.age_range = (age_range.get("Low"), age_range.get("High"))
        else:
            self.age_range = None
        self.smile = face.get("Smile", {}).get("Value")
        self.eyeglasses = face.get("Eyeglasses", {}).get("Value")
        self.sunglasses = face.get("Sunglasses", {}).get("Value")
        self.gender = face.get("Gender", {}).get("Value", None)
        self.beard = face.get("Beard", {}).get("Value")
        self.mustache = face.get("Mustache", {}).get("Value")
        self.eyes_open = face.get("EyesOpen", {}).get("Value")
        self.mouth_open = face.get("MouthOpen", {}).get("Value")
        self.emotions = [
            emo.get("Type")
            for emo in face.get("Emotions", [])
            if emo.get("Confidence", 0) > 50
        ]
        self.face_id = face.get("FaceId")
        self.image_id = face.get("ImageId")
        self.timestamp = timestamp

    def to_dict(self):
        """
        Renders some of the face data to a dict.

        :return: A dict that contains the face data.
        """
        rendering = {}
        if self.bounding_box is not None:
            rendering["bounding_box"] = self.bounding_box
        if self.age_range is not None:
            rendering["age"] = f"{self.age_range[0]} - {self.age_range[1]}"
        if self.gender is not None:
            rendering["gender"] = self.gender
        if self.emotions:
            rendering["emotions"] = self.emotions
        if self.face_id is not None:
            rendering["face_id"] = self.face_id
```

```
        if self.image_id is not None:
            rendering["image_id"] = self.image_id
        if self.timestamp is not None:
            rendering["timestamp"] = self.timestamp
        has = []
        if self.smile:
            has.append("smile")
        if self.eyeglasses:
            has.append("eyeglasses")
        if self.sunglasses:
            has.append("sunglasses")
        if self.beard:
            has.append("beard")
        if self.mustache:
            has.append("mustache")
        if self.eyes_open:
            has.append("open eyes")
        if self.mouth_open:
            has.append("open mouth")
        if has:
            rendering["has"] = has
        return rendering



class RekognitionCelebrity:
    """Encapsulates an Amazon Rekognition celebrity."""

    def __init__(self, celebrity, timestamp=None):
        """
        Initializes the celebrity object.

        :param celebrity: Celebrity data, in the format returned by Amazon
 Rekognition
                          functions.
        :param timestamp: The time when the celebrity was detected, if the
 celebrity
                          was detected in a video.
        """
        self.info_urls = celebrity.get("Urls")
        self.name = celebrity.get("Name")
        self.id = celebrity.get("Id")
        self.face = RekognitionFace(celebrity.get("Face"))
        self.confidence = celebrity.get("MatchConfidence")
```

```
        self.bounding_box = celebrity.get("BoundingBox")
        self.timestamp = timestamp

    def to_dict(self):
        """
        Renders some of the celebrity data to a dict.

        :return: A dict that contains the celebrity data.
        """
        rendering = self.face.to_dict()
        if self.name is not None:
            rendering["name"] = self.name
        if self.info_urls:
            rendering["info URLs"] = self.info_urls
        if self.timestamp is not None:
            rendering["timestamp"] = self.timestamp
        return rendering



class RekognitionPerson:
    """Encapsulates an Amazon Rekognition person."""

    def __init__(self, person, timestamp=None):
        """
        Initializes the person object.

        :param person: Person data, in the format returned by Amazon Rekognition
                       functions.
        :param timestamp: The time when the person was detected, if the person
                          was detected in a video.
        """
        self.index = person.get("Index")
        self.bounding_box = person.get("BoundingBox")
        face = person.get("Face")
        self.face = RekognitionFace(face) if face is not None else None
        self.timestamp = timestamp

    def to_dict(self):
        """
        Renders some of the person data to a dict.

        :return: A dict that contains the person data.
        """
```

```
        rendering = self.face.to_dict() if self.face is not None else {}
        if self.index is not None:
            rendering["index"] = self.index
        if self.bounding_box is not None:
            rendering["bounding_box"] = self.bounding_box
        if self.timestamp is not None:
            rendering["timestamp"] = self.timestamp
        return rendering



class RekognitionLabel:
    """Encapsulates an Amazon Rekognition label."""

    def __init__(self, label, timestamp=None):
        """
        Initializes the label object.

        :param label: Label data, in the format returned by Amazon Rekognition
                      functions.
        :param timestamp: The time when the label was detected, if the label
                          was detected in a video.
        """
        self.name = label.get("Name")
        self.confidence = label.get("Confidence")
        self.instances = label.get("Instances")
        self.parents = label.get("Parents")
        self.timestamp = timestamp

    def to_dict(self):
        """
        Renders some of the label data to a dict.

        :return: A dict that contains the label data.
        """
        rendering = {}
        if self.name is not None:
            rendering["name"] = self.name
        if self.timestamp is not None:
            rendering["timestamp"] = self.timestamp
        return rendering
```

```
class RekognitionModerationLabel:
    """Encapsulates an Amazon Rekognition moderation label."""

    def __init__(self, label, timestamp=None):
        """
        Initializes the moderation label object.

        :param label: Label data, in the format returned by Amazon Rekognition
                      functions.
        :param timestamp: The time when the moderation label was detected, if the
                          label was detected in a video.
        """
        self.name = label.get("Name")
        self.confidence = label.get("Confidence")
        self.parent_name = label.get("ParentName")
        self.timestamp = timestamp

    def to_dict(self):
        """
        Renders some of the moderation label data to a dict.

        :return: A dict that contains the moderation label data.
        """
        rendering = {}
        if self.name is not None:
            rendering["name"] = self.name
        if self.parent_name is not None:
            rendering["parent_name"] = self.parent_name
        if self.timestamp is not None:
            rendering["timestamp"] = self.timestamp
        return rendering



class RekognitionText:
    """Encapsulates an Amazon Rekognition text element."""

    def __init__(self, text_data):
        """
        Initializes the text object.

        :param text_data: Text data, in the format returned by Amazon Rekognition
                          functions.
        """
```

```
        self.text = text_data.get("DetectedText")
        self.kind = text_data.get("Type")
        self.id = text_data.get("Id")
        self.parent_id = text_data.get("ParentId")
        self.confidence = text_data.get("Confidence")
        self.geometry = text_data.get("Geometry")

    def to_dict(self):
        """
        Renders some of the text data to a dict.

        :return: A dict that contains the text data.
        """
        rendering = {}
        if self.text is not None:
            rendering["text"] = self.text
        if self.kind is not None:
            rendering["kind"] = self.kind
        if self.geometry is not None:
            rendering["polygon"] = self.geometry.get("Polygon")
        return rendering
```

Utilice las clases contenedoras para detectar elementos en las imágenes y mostrar sus cuadros delimitadores. Las imágenes utilizadas en este ejemplo se encuentran GitHub junto con las instrucciones y más código.

```
def usage_demo():
    print("-" * 88)
    print("Welcome to the Amazon Rekognition image detection demo!")
    print("-" * 88)

    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")
    rekognition_client = boto3.client("rekognition")
    street_scene_file_name = ".media/pexels-kaique-rocha-109919.jpg"
    celebrity_file_name = ".media/pexels-pixabay-53370.jpg"
    one_girl_url = "https://dhei5unw3vrsx.cloudfront.net/images/
source3_resized.jpg"
    three_girls_url = "https://dhei5unw3vrsx.cloudfront.net/images/
target3_resized.jpg"
    swimwear_object = boto3.resource("s3").Object(
```

```
        "console-sample-images-pdx", "yoga_swimwear.jpg"
    )
    book_file_name = ".media/pexels-christina-morillo-1181671.jpg"

    street_scene_image = RekognitionImage.from_file(
        street_scene_file_name, rekognition_client
    )
    print(f"Detecting faces in {street_scene_image.image_name}...")
    faces = street_scene_image.detect_faces()
    print(f"Found {len(faces)} faces, here are the first three.")
    for face in faces[:3]:
        pprint(face.to_dict())
    show_bounding_boxes(
        street_scene_image.image["Bytes"],
        [[face.bounding_box for face in faces]],
        ["aqua"],
    )
    input("Press Enter to continue.")

    print(f"Detecting labels in {street_scene_image.image_name}...")
    labels = street_scene_image.detect_labels(100)
    print(f"Found {len(labels)} labels.")
    for label in labels:
        pprint(label.to_dict())
    names = []
    box_sets = []
    colors = ["aqua", "red", "white", "blue", "yellow", "green"]
    for label in labels:
        if label.instances:
            names.append(label.name)
            box_sets.append([inst["BoundingBox"] for inst in label.instances])
    print(f"Showing bounding boxes for {names} in {colors[:len(names)]}.")
    show_bounding_boxes(
        street_scene_image.image["Bytes"], box_sets, colors[: len(names)]
    )
    input("Press Enter to continue.")

    celebrity_image = RekognitionImage.from_file(
        celebrity_file_name, rekognition_client
    )
    print(f"Detecting celebrities in {celebrity_image.image_name}...")
    celebs, others = celebrity_image.recognize_celebrities()
    print(f"Found {len(celebs)} celebrities.")
    for celeb in celebs:
```

```
        pprint(celeb.to_dict())
    show_bounding_boxes(
        celebrity_image.image["Bytes"],
        [[celeb.face.bounding_box for celeb in celebs]],
        ["aqua"],
    )
    input("Press Enter to continue.")

    girl_image_response = requests.get(one_girl_url)
    girl_image = RekognitionImage(
        {"Bytes": girl_image_response.content}, "one-girl", rekognition_client
    )
    group_image_response = requests.get(three_girls_url)
    group_image = RekognitionImage(
        {"Bytes": group_image_response.content}, "three-girls",
rekognition_client
    )
    print("Comparing reference face to group of faces...")
    matches, unmatches = girl_image.compare_faces(group_image, 80)
    print(f"Found {len(matches)} face matching the reference face.")
    show_bounding_boxes(
        group_image.image["Bytes"],
        [[match.bounding_box for match in matches]],
        ["aqua"],
    )
    input("Press Enter to continue.")

    swimwear_image = RekognitionImage.from_bucket(swimwear_object,
rekognition_client)
    print(f"Detecting suggestive content in {swimwear_object.key}...")
    labels = swimwear_image.detect_moderation_labels()
    print(f"Found {len(labels)} moderation labels.")
    for label in labels:
        pprint(label.to_dict())
    input("Press Enter to continue.")

    book_image = RekognitionImage.from_file(book_file_name, rekognition_client)
    print(f"Detecting text in {book_image.image_name}...")
    texts = book_image.detect_text()
    print(f"Found {len(texts)} text instances. Here are the first seven:")
    for text in texts[:7]:
        pprint(text.to_dict())
    show_polygons(
```

```
        book_image.image["Bytes"], [text.geometry["Polygon"] for text in texts],
"aqua"
    )

    print("Thanks for watching!")
    print("-" * 88)
```

Para obtener una lista completa de las guías para desarrolladores del AWS SDK y ejemplos de código, consulteUso de Rekognition con un SDK AWS. En este tema también se incluye información sobre cómo comenzar a utilizar el SDK y detalles sobre sus versiones anteriores.

Detecta rostros en una imagen mediante un AWS SDK

En el siguiente ejemplo de código, se muestra cómo:

- Guarde una imagen en un bucket de Amazon S3.
- Utilice Amazon Rekognition para detectar información faciales, como el rango de edad, el género y las emociones (por ejemplo, una sonrisa).
- Muestre esos detalles.

Rust

SDK para Rust

Guarde la imagen en un bucket de Amazon S3 con el prefijo uploads, use Amazon Rekognition para detectar información faciales, como el rango de edad, el género y las emociones (por ejemplo, una sonrisa) y muestre esos detalles.

Para ver el código fuente completo y las instrucciones sobre cómo configurarlo y ejecutarlo, consulta el ejemplo completo en GitHub.

Servicios utilizados en este ejemplo

- Amazon Rekognition
- Amazon S3

Para obtener una lista completa de guías para desarrolladores del AWS SDK y ejemplos de código, consulteUso de Rekognition con un SDK AWS. En este tema también se incluye información sobre cómo comenzar a utilizar el SDK y detalles sobre sus versiones anteriores.

Detecte información en vídeos con Amazon Rekognition y el SDK AWS

En el siguiente ejemplo de código, se muestra cómo:

- Inicie Amazon Rekognition Jobs para detectar elementos como personas, objetos y texto en los vídeos.
- Compruebe el estado de los trabajos hasta que se terminan.
- Obtener la lista de elementos detectados por cada trabajo.

Java

SDK para Java 2.x

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

Obtenga información sobre famosos a partir de un vídeo ubicado en un bucket de Amazon S3.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.S3Object;
import software.amazon.awssdk.services.rekognition.model.NotificationChannel;
import software.amazon.awssdk.services.rekognition.model.Video;
import
 software.amazon.awssdk.services.rekognition.model.StartCelebrityRecognitionResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import
 software.amazon.awssdk.services.rekognition.model.CelebrityRecognitionSortBy;
import software.amazon.awssdk.services.rekognition.model.VideoMetadata;
import software.amazon.awssdk.services.rekognition.model.CelebrityRecognition;
import software.amazon.awssdk.services.rekognition.model.CelebrityDetail;
import
 software.amazon.awssdk.services.rekognition.model.StartCelebrityRecognitionRequest;
```

```
import
 software.amazon.awssdk.services.rekognition.model.GetCelebrityRecognitionRequest;
import
 software.amazon.awssdk.services.rekognition.model.GetCelebrityRecognitionResponse;
import java.util.List;

/**
 * To run this code example, ensure that you perform the Prerequisites as stated
 * in the Amazon Rekognition Guide:
 * https://docs.aws.amazon.com/rekognition/latest/dg/video-analyzing-with-
sqs.html
 *
 * Also, ensure that set up your development environment, including your
 * credentials.
 *
 * For information, see this documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */

public class VideoCelebrityDetection {
    private static String startJobId = "";

    public static void main(String[] args) {
        final String usage = """

                Usage:    <bucket> <video> <topicArn> <roleArn>

                Where:
                   bucket - The name of the bucket in which the video is located
 (for example, (for example, myBucket).\s
                   video - The name of video (for example, people.mp4).\s
                   topicArn - The ARN of the Amazon Simple Notification Service
 (Amazon SNS) topic.\s
                   roleArn - The ARN of the AWS Identity and Access Management
 (IAM) role to use.\s
                """;

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

```
        String bucket = args[0];
        String video = args[1];
        String topicArn = args[2];
        String roleArn = args[3];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        NotificationChannel channel = NotificationChannel.builder()
                .snsTopicArn(topicArn)
                .roleArn(roleArn)
                .build();

        startCelebrityDetection(rekClient, channel, bucket, video);
        getCelebrityDetectionResults(rekClient);
        System.out.println("This example is done!");
        rekClient.close();
    }

    public static void startCelebrityDetection(RekognitionClient rekClient,
            NotificationChannel channel,
            String bucket,
            String video) {
        try {
            S3Object s3Obj = S3Object.builder()
                    .bucket(bucket)
                    .name(video)
                    .build();

            Video vidOb = Video.builder()
                    .s3Object(s3Obj)
                    .build();

            StartCelebrityRecognitionRequest recognitionRequest =
StartCelebrityRecognitionRequest.builder()
                    .jobTag("Celebrities")
                    .notificationChannel(channel)
                    .video(vidOb)
                    .build();

            StartCelebrityRecognitionResponse startCelebrityRecognitionResult =
rekClient
                    .startCelebrityRecognition(recognitionRequest);
```

```
            startJobId = startCelebrityRecognitionResult.jobId();

        } catch (RekognitionException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void getCelebrityDetectionResults(RekognitionClient rekClient)
{
        try {
            String paginationToken = null;
            GetCelebrityRecognitionResponse recognitionResponse = null;
            boolean finished = false;
            String status;
            int yy = 0;

            do {
                if (recognitionResponse != null)
                    paginationToken = recognitionResponse.nextToken();

                GetCelebrityRecognitionRequest recognitionRequest =
GetCelebrityRecognitionRequest.builder()
                        .jobId(startJobId)
                        .nextToken(paginationToken)
                        .sortBy(CelebrityRecognitionSortBy.TIMESTAMP)
                        .maxResults(10)
                        .build();

                // Wait until the job succeeds
                while (!finished) {
                    recognitionResponse =
rekClient.getCelebrityRecognition(recognitionRequest);
                    status = recognitionResponse.jobStatusAsString();

                    if (status.compareTo("SUCCEEDED") == 0)
                        finished = true;
                    else {
                        System.out.println(yy + " status is: " + status);
                        Thread.sleep(1000);
                    }
                    yy++;
                }
```

```
                finished = false;

                // Proceed when the job is done - otherwise VideoMetadata is
 null.
                VideoMetadata videoMetaData =
 recognitionResponse.videoMetadata();
                System.out.println("Format: " + videoMetaData.format());
                System.out.println("Codec: " + videoMetaData.codec());
                System.out.println("Duration: " +
 videoMetaData.durationMillis());
                System.out.println("FrameRate: " + videoMetaData.frameRate());
                System.out.println("Job");

                List<CelebrityRecognition> celebs =
 recognitionResponse.celebrities();
                for (CelebrityRecognition celeb : celebs) {
                    long seconds = celeb.timestamp() / 1000;
                    System.out.print("Sec: " + seconds + " ");
                    CelebrityDetail details = celeb.celebrity();
                    System.out.println("Name: " + details.name());
                    System.out.println("Id: " + details.id());
                    System.out.println();
                }

            } while (recognitionResponse.nextToken() != null);

        } catch (RekognitionException | InterruptedException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

Detecte etiquetas en un vídeo mediante una operación de detección de etiquetas.

```
import com.fasterxml.jackson.core.JsonProcessingException;
import com.fasterxml.jackson.databind.JsonMappingException;
import com.fasterxml.jackson.databind.JsonNode;
import com.fasterxml.jackson.databind.ObjectMapper;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
```

```
import
 software.amazon.awssdk.services.rekognition.model.StartLabelDetectionResponse;
import software.amazon.awssdk.services.rekognition.model.NotificationChannel;
import software.amazon.awssdk.services.rekognition.model.S3Object;
import software.amazon.awssdk.services.rekognition.model.Video;
import
 software.amazon.awssdk.services.rekognition.model.StartLabelDetectionRequest;
import
 software.amazon.awssdk.services.rekognition.model.GetLabelDetectionRequest;
import
 software.amazon.awssdk.services.rekognition.model.GetLabelDetectionResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import software.amazon.awssdk.services.rekognition.model.LabelDetectionSortBy;
import software.amazon.awssdk.services.rekognition.model.VideoMetadata;
import software.amazon.awssdk.services.rekognition.model.LabelDetection;
import software.amazon.awssdk.services.rekognition.model.Label;
import software.amazon.awssdk.services.rekognition.model.Instance;
import software.amazon.awssdk.services.rekognition.model.Parent;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.Message;
import software.amazon.awssdk.services.sqs.model.ReceiveMessageRequest;
import software.amazon.awssdk.services.sqs.model.DeleteMessageRequest;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class VideoDetect {
    private static String startJobId = "";

    public static void main(String[] args) {
        final String usage = """

                Usage:    <bucket> <video> <queueUrl> <topicArn> <roleArn>

                Where:
                   bucket - The name of the bucket in which the video is located
 (for example, (for example, myBucket).\s
```

```
                video - The name of the video (for example, people.mp4).\s
                queueUrl- The URL of a SQS queue.\s
                topicArn - The ARN of the Amazon Simple Notification Service
(Amazon SNS) topic.\s
                roleArn - The ARN of the AWS Identity and Access Management
(IAM) role to use.\s
                """;

        if (args.length != 5) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucket = args[0];
        String video = args[1];
        String queueUrl = args[2];
        String topicArn = args[3];
        String roleArn = args[4];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        SqsClient sqs = SqsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        NotificationChannel channel = NotificationChannel.builder()
                .snsTopicArn(topicArn)
                .roleArn(roleArn)
                .build();

        startLabels(rekClient, channel, bucket, video);
        getLabelJob(rekClient, sqs, queueUrl);
        System.out.println("This example is done!");
        sqs.close();
        rekClient.close();
    }

    public static void startLabels(RekognitionClient rekClient,
            NotificationChannel channel,
            String bucket,
            String video) {
        try {
```

```
            S3Object s3Obj = S3Object.builder()
                    .bucket(bucket)
                    .name(video)
                    .build();

            Video vidOb = Video.builder()
                    .s3Object(s3Obj)
                    .build();

            StartLabelDetectionRequest labelDetectionRequest =
StartLabelDetectionRequest.builder()
                    .jobTag("DetectingLabels")
                    .notificationChannel(channel)
                    .video(vidOb)
                    .minConfidence(50F)
                    .build();

            StartLabelDetectionResponse labelDetectionResponse =
rekClient.startLabelDetection(labelDetectionRequest);
            startJobId = labelDetectionResponse.jobId();

            boolean ans = true;
            String status = "";
            int yy = 0;
            while (ans) {

                GetLabelDetectionRequest detectionRequest =
GetLabelDetectionRequest.builder()
                        .jobId(startJobId)
                        .maxResults(10)
                        .build();

                GetLabelDetectionResponse result =
rekClient.getLabelDetection(detectionRequest);
                status = result.jobStatusAsString();

                if (status.compareTo("SUCCEEDED") == 0)
                    ans = false;
                else
                    System.out.println(yy + " status is: " + status);

                Thread.sleep(1000);
                yy++;
            }
```

```
            System.out.println(startJobId + " status is: " + status);

        } catch (RekognitionException | InterruptedException e) {
            e.getMessage();
            System.exit(1);
        }
    }

    public static void getLabelJob(RekognitionClient rekClient, SqsClient sqs,
String queueUrl) {
        List<Message> messages;
        ReceiveMessageRequest messageRequest = ReceiveMessageRequest.builder()
                .queueUrl(queueUrl)
                .build();

        try {
            messages = sqs.receiveMessage(messageRequest).messages();

            if (!messages.isEmpty()) {
                for (Message message : messages) {
                    String notification = message.body();

                    // Get the status and job id from the notification
                    ObjectMapper mapper = new ObjectMapper();
                    JsonNode jsonMessageTree = mapper.readTree(notification);
                    JsonNode messageBodyText = jsonMessageTree.get("Message");
                    ObjectMapper operationResultMapper = new ObjectMapper();
                    JsonNode jsonResultTree =
operationResultMapper.readTree(messageBodyText.textValue());
                    JsonNode operationJobId = jsonResultTree.get("JobId");
                    JsonNode operationStatus = jsonResultTree.get("Status");
                    System.out.println("Job found in JSON is " + operationJobId);

                    DeleteMessageRequest deleteMessageRequest =
DeleteMessageRequest.builder()
                            .queueUrl(queueUrl)
                            .build();

                    String jobId = operationJobId.textValue();
                    if (startJobId.compareTo(jobId) == 0) {
                        System.out.println("Job id: " + operationJobId);
                        System.out.println("Status : " +
operationStatus.toString());
```

```
                        if (operationStatus.asText().equals("SUCCEEDED"))
                            getResultsLabels(rekClient);
                        else
                            System.out.println("Video analysis failed");

                        sqs.deleteMessage(deleteMessageRequest);
                    } else {
                        System.out.println("Job received was not job " +
startJobId);
                        sqs.deleteMessage(deleteMessageRequest);
                    }
                }
            }

        } catch (RekognitionException e) {
            e.getMessage();
            System.exit(1);
        } catch (JsonMappingException e) {
            e.printStackTrace();
        } catch (JsonProcessingException e) {
            e.printStackTrace();
        }
    }

    // Gets the job results by calling GetLabelDetection
    private static void getResultsLabels(RekognitionClient rekClient) {

        int maxResults = 10;
        String paginationToken = null;
        GetLabelDetectionResponse labelDetectionResult = null;

        try {
            do {
                if (labelDetectionResult != null)
                    paginationToken = labelDetectionResult.nextToken();

                GetLabelDetectionRequest labelDetectionRequest =
GetLabelDetectionRequest.builder()
                        .jobId(startJobId)
                        .sortBy(LabelDetectionSortBy.TIMESTAMP)
                        .maxResults(maxResults)
                        .nextToken(paginationToken)
                        .build();
```

```
                labelDetectionResult =
rekClient.getLabelDetection(labelDetectionRequest);
                VideoMetadata videoMetaData =
labelDetectionResult.videoMetadata();
                System.out.println("Format: " + videoMetaData.format());
                System.out.println("Codec: " + videoMetaData.codec());
                System.out.println("Duration: " +
videoMetaData.durationMillis());
                System.out.println("FrameRate: " + videoMetaData.frameRate());

                List<LabelDetection> detectedLabels =
labelDetectionResult.labels();
                for (LabelDetection detectedLabel : detectedLabels) {
                    long seconds = detectedLabel.timestamp();
                    Label label = detectedLabel.label();
                    System.out.println("Millisecond: " + seconds + " ");

                    System.out.println("   Label:" + label.name());
                    System.out.println("   Confidence:" +
detectedLabel.label().confidence().toString());

                    List<Instance> instances = label.instances();
                    System.out.println("   Instances of " + label.name());

                    if (instances.isEmpty()) {
                        System.out.println("        " + "None");
                    } else {
                        for (Instance instance : instances) {
                            System.out.println("        Confidence: " +
instance.confidence().toString());
                            System.out.println("        Bounding box: " +
instance.boundingBox().toString());
                        }
                    }
                    System.out.println("   Parent labels for " + label.name() +
":");
                    List<Parent> parents = label.parents();

                    if (parents.isEmpty()) {
                        System.out.println("        None");
                    } else {
                        for (Parent parent : parents) {
                            System.out.println("   " + parent.name());
```

```
                        }
                    }
                    System.out.println();
                }
            } while (labelDetectionResult != null &&
 labelDetectionResult.nextToken() != null);

        } catch (RekognitionException e) {
            e.getMessage();
            System.exit(1);
        }
    }
}
```

Detecte rostros en un vídeo almacenado en un bucket de Amazon S3.

```
import com.fasterxml.jackson.core.JsonProcessingException;
import com.fasterxml.jackson.databind.JsonMappingException;
import com.fasterxml.jackson.databind.JsonNode;
import com.fasterxml.jackson.databind.ObjectMapper;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import
 software.amazon.awssdk.services.rekognition.model.StartLabelDetectionResponse;
import software.amazon.awssdk.services.rekognition.model.NotificationChannel;
import software.amazon.awssdk.services.rekognition.model.S3Object;
import software.amazon.awssdk.services.rekognition.model.Video;
import
 software.amazon.awssdk.services.rekognition.model.StartLabelDetectionRequest;
import
 software.amazon.awssdk.services.rekognition.model.GetLabelDetectionRequest;
import
 software.amazon.awssdk.services.rekognition.model.GetLabelDetectionResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import software.amazon.awssdk.services.rekognition.model.LabelDetectionSortBy;
import software.amazon.awssdk.services.rekognition.model.VideoMetadata;
import software.amazon.awssdk.services.rekognition.model.LabelDetection;
import software.amazon.awssdk.services.rekognition.model.Label;
import software.amazon.awssdk.services.rekognition.model.Instance;
import software.amazon.awssdk.services.rekognition.model.Parent;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.Message;
```

```
import software.amazon.awssdk.services.sqs.model.ReceiveMessageRequest;
import software.amazon.awssdk.services.sqs.model.DeleteMessageRequest;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class VideoDetect {
    private static String startJobId = "";

    public static void main(String[] args) {
        final String usage = """

                Usage:    <bucket> <video> <queueUrl> <topicArn> <roleArn>

                Where:
                   bucket - The name of the bucket in which the video is located
 (for example, (for example, myBucket).\s
                   video - The name of the video (for example, people.mp4).\s
                   queueUrl- The URL of a SQS queue.\s
                   topicArn - The ARN of the Amazon Simple Notification Service
 (Amazon SNS) topic.\s
                   roleArn - The ARN of the AWS Identity and Access Management
 (IAM) role to use.\s
                """;

        if (args.length != 5) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucket = args[0];
        String video = args[1];
        String queueUrl = args[2];
        String topicArn = args[3];
        String roleArn = args[4];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
```

```
                .region(region)
                .build();

        SqsClient sqs = SqsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        NotificationChannel channel = NotificationChannel.builder()
                .snsTopicArn(topicArn)
                .roleArn(roleArn)
                .build();

        startLabels(rekClient, channel, bucket, video);
        getLabelJob(rekClient, sqs, queueUrl);
        System.out.println("This example is done!");
        sqs.close();
        rekClient.close();
    }

    public static void startLabels(RekognitionClient rekClient,
            NotificationChannel channel,
            String bucket,
            String video) {
        try {
            S3Object s3Obj = S3Object.builder()
                    .bucket(bucket)
                    .name(video)
                    .build();

            Video vidOb = Video.builder()
                    .s3Object(s3Obj)
                    .build();

            StartLabelDetectionRequest labelDetectionRequest =
StartLabelDetectionRequest.builder()
                    .jobTag("DetectingLabels")
                    .notificationChannel(channel)
                    .video(vidOb)
                    .minConfidence(50F)
                    .build();

            StartLabelDetectionResponse labelDetectionResponse =
rekClient.startLabelDetection(labelDetectionRequest);
            startJobId = labelDetectionResponse.jobId();
```

```
            boolean ans = true;
            String status = "";
            int yy = 0;
            while (ans) {

                GetLabelDetectionRequest detectionRequest =
GetLabelDetectionRequest.builder()
                        .jobId(startJobId)
                        .maxResults(10)
                        .build();

                GetLabelDetectionResponse result =
rekClient.getLabelDetection(detectionRequest);
                status = result.jobStatusAsString();

                if (status.compareTo("SUCCEEDED") == 0)
                    ans = false;
                else
                    System.out.println(yy + " status is: " + status);

                Thread.sleep(1000);
                yy++;
            }

            System.out.println(startJobId + " status is: " + status);

        } catch (RekognitionException | InterruptedException e) {
            e.getMessage();
            System.exit(1);
        }
    }

    public static void getLabelJob(RekognitionClient rekClient, SqsClient sqs,
String queueUrl) {
        List<Message> messages;
        ReceiveMessageRequest messageRequest = ReceiveMessageRequest.builder()
                .queueUrl(queueUrl)
                .build();

        try {
            messages = sqs.receiveMessage(messageRequest).messages();

            if (!messages.isEmpty()) {
```

```
                for (Message message : messages) {
                    String notification = message.body();

                    // Get the status and job id from the notification
                    ObjectMapper mapper = new ObjectMapper();
                    JsonNode jsonMessageTree = mapper.readTree(notification);
                    JsonNode messageBodyText = jsonMessageTree.get("Message");
                    ObjectMapper operationResultMapper = new ObjectMapper();
                    JsonNode jsonResultTree =
operationResultMapper.readTree(messageBodyText.textValue());
                    JsonNode operationJobId = jsonResultTree.get("JobId");
                    JsonNode operationStatus = jsonResultTree.get("Status");
                    System.out.println("Job found in JSON is " + operationJobId);

                    DeleteMessageRequest deleteMessageRequest =
DeleteMessageRequest.builder()
                            .queueUrl(queueUrl)
                            .build();

                    String jobId = operationJobId.textValue();
                    if (startJobId.compareTo(jobId) == 0) {
                        System.out.println("Job id: " + operationJobId);
                        System.out.println("Status : " +
operationStatus.toString());

                        if (operationStatus.asText().equals("SUCCEEDED"))
                            getResultsLabels(rekClient);
                        else
                            System.out.println("Video analysis failed");

                        sqs.deleteMessage(deleteMessageRequest);
                    } else {
                        System.out.println("Job received was not job " +
startJobId);
                        sqs.deleteMessage(deleteMessageRequest);
                    }
                }
            }

        } catch (RekognitionException e) {
            e.getMessage();
            System.exit(1);
        } catch (JsonMappingException e) {
            e.printStackTrace();
```

```
        } catch (JsonProcessingException e) {
            e.printStackTrace();
        }
    }

    // Gets the job results by calling GetLabelDetection
    private static void getResultsLabels(RekognitionClient rekClient) {

        int maxResults = 10;
        String paginationToken = null;
        GetLabelDetectionResponse labelDetectionResult = null;

        try {
            do {
                if (labelDetectionResult != null)
                    paginationToken = labelDetectionResult.nextToken();

                GetLabelDetectionRequest labelDetectionRequest =
GetLabelDetectionRequest.builder()
                        .jobId(startJobId)
                        .sortBy(LabelDetectionSortBy.TIMESTAMP)
                        .maxResults(maxResults)
                        .nextToken(paginationToken)
                        .build();

                labelDetectionResult =
rekClient.getLabelDetection(labelDetectionRequest);
                VideoMetadata videoMetaData =
labelDetectionResult.videoMetadata();
                System.out.println("Format: " + videoMetaData.format());
                System.out.println("Codec: " + videoMetaData.codec());
                System.out.println("Duration: " +
videoMetaData.durationMillis());
                System.out.println("FrameRate: " + videoMetaData.frameRate());

                List<LabelDetection> detectedLabels =
labelDetectionResult.labels();
                for (LabelDetection detectedLabel : detectedLabels) {
                    long seconds = detectedLabel.timestamp();
                    Label label = detectedLabel.label();
                    System.out.println("Millisecond: " + seconds + " ");

                    System.out.println("   Label:" + label.name());
```

```
                    System.out.println("   Confidence:" +
 detectedLabel.label().confidence().toString());

                    List<Instance> instances = label.instances();
                    System.out.println("   Instances of " + label.name());

                    if (instances.isEmpty()) {
                        System.out.println("        " + "None");
                    } else {
                        for (Instance instance : instances) {
                            System.out.println("        Confidence: " +
 instance.confidence().toString());
                            System.out.println("        Bounding box: " +
 instance.boundingBox().toString());
                        }
                    }
                    System.out.println("   Parent labels for " + label.name() +
 ":");
                    List<Parent> parents = label.parents();

                    if (parents.isEmpty()) {
                        System.out.println("        None");
                    } else {
                        for (Parent parent : parents) {
                            System.out.println("   " + parent.name());
                        }
                    }
                    System.out.println();
                }
            } while (labelDetectionResult != null &&
 labelDetectionResult.nextToken() != null);

        } catch (RekognitionException e) {
            e.getMessage();
            System.exit(1);
        }
    }
}
```

Detecte contenido inapropiado u ofensivo en un vídeo almacenado en un bucket de Amazon S3.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.NotificationChannel;
import software.amazon.awssdk.services.rekognition.model.S3Object;
import software.amazon.awssdk.services.rekognition.model.Video;
import
 software.amazon.awssdk.services.rekognition.model.StartContentModerationRequest;
import
 software.amazon.awssdk.services.rekognition.model.StartContentModerationResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import
 software.amazon.awssdk.services.rekognition.model.GetContentModerationResponse;
import
 software.amazon.awssdk.services.rekognition.model.GetContentModerationRequest;
import software.amazon.awssdk.services.rekognition.model.VideoMetadata;
import
 software.amazon.awssdk.services.rekognition.model.ContentModerationDetection;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class VideoDetectInappropriate {
    private static String startJobId = "";

    public static void main(String[] args) {

        final String usage = """

                Usage:    <bucket> <video> <topicArn> <roleArn>

                Where:
                   bucket - The name of the bucket in which the video is located
 (for example, (for example, myBucket).\s
                   video - The name of video (for example, people.mp4).\s
                   topicArn - The ARN of the Amazon Simple Notification Service
 (Amazon SNS) topic.\s
```

```
                    roleArn - The ARN of the AWS Identity and Access Management
(IAM) role to use.\s
                """;

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

        String bucket = args[0];
        String video = args[1];
        String topicArn = args[2];
        String roleArn = args[3];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        NotificationChannel channel = NotificationChannel.builder()
                .snsTopicArn(topicArn)
                .roleArn(roleArn)
                .build();

        startModerationDetection(rekClient, channel, bucket, video);
        getModResults(rekClient);
        System.out.println("This example is done!");
        rekClient.close();
    }

    public static void startModerationDetection(RekognitionClient rekClient,
            NotificationChannel channel,
            String bucket,
            String video) {

        try {
            S3Object s3Obj = S3Object.builder()
                    .bucket(bucket)
                    .name(video)
                    .build();

            Video vidOb = Video.builder()
                    .s3Object(s3Obj)
                    .build();
```

```
            StartContentModerationRequest modDetectionRequest =
StartContentModerationRequest.builder()
                    .jobTag("Moderation")
                    .notificationChannel(channel)
                    .video(vidOb)
                    .build();

            StartContentModerationResponse startModDetectionResult = rekClient
                    .startContentModeration(modDetectionRequest);
            startJobId = startModDetectionResult.jobId();

        } catch (RekognitionException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void getModResults(RekognitionClient rekClient) {
        try {
            String paginationToken = null;
            GetContentModerationResponse modDetectionResponse = null;
            boolean finished = false;
            String status;
            int yy = 0;

            do {
                if (modDetectionResponse != null)
                    paginationToken = modDetectionResponse.nextToken();

                GetContentModerationRequest modRequest =
GetContentModerationRequest.builder()
                        .jobId(startJobId)
                        .nextToken(paginationToken)
                        .maxResults(10)
                        .build();

                // Wait until the job succeeds.
                while (!finished) {
                    modDetectionResponse =
rekClient.getContentModeration(modRequest);
                    status = modDetectionResponse.jobStatusAsString();

                    if (status.compareTo("SUCCEEDED") == 0)
                        finished = true;
```

```
                    else {
                        System.out.println(yy + " status is: " + status);
                        Thread.sleep(1000);
                    }
                    yy++;
                }

                finished = false;

                // Proceed when the job is done - otherwise VideoMetadata is
 null.
                VideoMetadata videoMetaData =
 modDetectionResponse.videoMetadata();
                System.out.println("Format: " + videoMetaData.format());
                System.out.println("Codec: " + videoMetaData.codec());
                System.out.println("Duration: " +
 videoMetaData.durationMillis());
                System.out.println("FrameRate: " + videoMetaData.frameRate());
                System.out.println("Job");

                List<ContentModerationDetection> mods =
 modDetectionResponse.moderationLabels();
                for (ContentModerationDetection mod : mods) {
                    long seconds = mod.timestamp() / 1000;
                    System.out.print("Mod label: " + seconds + " ");
                    System.out.println(mod.moderationLabel().toString());
                    System.out.println();
                }

            } while (modDetectionResponse != null &&
 modDetectionResponse.nextToken() != null);

        } catch (RekognitionException | InterruptedException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

Detecte segmentos de señales técnicas y segmentos de detección de tomas en un vídeo almacenado en un bucket de Amazon S3.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.S3Object;
import software.amazon.awssdk.services.rekognition.model.NotificationChannel;
import software.amazon.awssdk.services.rekognition.model.Video;
import
 software.amazon.awssdk.services.rekognition.model.StartShotDetectionFilter;
import
 software.amazon.awssdk.services.rekognition.model.StartTechnicalCueDetectionFilter;
import
 software.amazon.awssdk.services.rekognition.model.StartSegmentDetectionFilters;
import
 software.amazon.awssdk.services.rekognition.model.StartSegmentDetectionRequest;
import
 software.amazon.awssdk.services.rekognition.model.StartSegmentDetectionResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import
 software.amazon.awssdk.services.rekognition.model.GetSegmentDetectionResponse;
import
 software.amazon.awssdk.services.rekognition.model.GetSegmentDetectionRequest;
import software.amazon.awssdk.services.rekognition.model.VideoMetadata;
import software.amazon.awssdk.services.rekognition.model.SegmentDetection;
import software.amazon.awssdk.services.rekognition.model.TechnicalCueSegment;
import software.amazon.awssdk.services.rekognition.model.ShotSegment;
import software.amazon.awssdk.services.rekognition.model.SegmentType;
import software.amazon.awssdk.services.sqs.SqsClient;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class VideoDetectSegment {
    private static String startJobId = "";

    public static void main(String[] args) {
        final String usage = """
```

```
                Usage:    <bucket> <video> <topicArn> <roleArn>

                Where:
                   bucket - The name of the bucket in which the video is located
(for example, (for example, myBucket).\s
                   video - The name of video (for example, people.mp4).\s
                   topicArn - The ARN of the Amazon Simple Notification Service
(Amazon SNS) topic.\s
                   roleArn - The ARN of the AWS Identity and Access Management
(IAM) role to use.\s
                """;

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

       String bucket = args[0];
       String video = args[1];
       String topicArn = args[2];
       String roleArn = args[3];

       Region region = Region.US_EAST_1;
       RekognitionClient rekClient = RekognitionClient.builder()
               .region(region)
               .build();

       SqsClient sqs = SqsClient.builder()
               .region(Region.US_EAST_1)
               .build();

       NotificationChannel channel = NotificationChannel.builder()
               .snsTopicArn(topicArn)
               .roleArn(roleArn)
               .build();

       startSegmentDetection(rekClient, channel, bucket, video);
       getSegmentResults(rekClient);
       System.out.println("This example is done!");
       sqs.close();
       rekClient.close();
   }

   public static void startSegmentDetection(RekognitionClient rekClient,
```

```
            NotificationChannel channel,
            String bucket,
            String video) {
        try {
            S3Object s3Obj = S3Object.builder()
                    .bucket(bucket)
                    .name(video)
                    .build();

            Video vidOb = Video.builder()
                    .s3Object(s3Obj)
                    .build();

            StartShotDetectionFilter cueDetectionFilter =
StartShotDetectionFilter.builder()
                    .minSegmentConfidence(60F)
                    .build();

            StartTechnicalCueDetectionFilter technicalCueDetectionFilter =
StartTechnicalCueDetectionFilter.builder()
                    .minSegmentConfidence(60F)
                    .build();

            StartSegmentDetectionFilters filters =
StartSegmentDetectionFilters.builder()
                    .shotFilter(cueDetectionFilter)
                    .technicalCueFilter(technicalCueDetectionFilter)
                    .build();

            StartSegmentDetectionRequest segDetectionRequest =
StartSegmentDetectionRequest.builder()
                    .jobTag("DetectingLabels")
                    .notificationChannel(channel)
                    .segmentTypes(SegmentType.TECHNICAL_CUE, SegmentType.SHOT)
                    .video(vidOb)
                    .filters(filters)
                    .build();

            StartSegmentDetectionResponse segDetectionResponse =
rekClient.startSegmentDetection(segDetectionRequest);
            startJobId = segDetectionResponse.jobId();

        } catch (RekognitionException e) {
            e.getMessage();
```

```
            System.exit(1);
        }
    }

    public static void getSegmentResults(RekognitionClient rekClient) {
        try {
            String paginationToken = null;
            GetSegmentDetectionResponse segDetectionResponse = null;
            boolean finished = false;
            String status;
            int yy = 0;

            do {
                if (segDetectionResponse != null)
                    paginationToken = segDetectionResponse.nextToken();

                GetSegmentDetectionRequest recognitionRequest =
GetSegmentDetectionRequest.builder()
                        .jobId(startJobId)
                        .nextToken(paginationToken)
                        .maxResults(10)
                        .build();

                // Wait until the job succeeds.
                while (!finished) {
                    segDetectionResponse =
rekClient.getSegmentDetection(recognitionRequest);
                    status = segDetectionResponse.jobStatusAsString();

                    if (status.compareTo("SUCCEEDED") == 0)
                        finished = true;
                    else {
                        System.out.println(yy + " status is: " + status);
                        Thread.sleep(1000);
                    }
                    yy++;
                }
                finished = false;

                // Proceed when the job is done - otherwise VideoMetadata is
null.
                List<VideoMetadata> videoMetaData =
segDetectionResponse.videoMetadata();
                for (VideoMetadata metaData : videoMetaData) {
```

```
                    System.out.println("Format: " + metaData.format());
                    System.out.println("Codec: " + metaData.codec());
                    System.out.println("Duration: " + metaData.durationMillis());
                    System.out.println("FrameRate: " + metaData.frameRate());
                    System.out.println("Job");
                }

                List<SegmentDetection> detectedSegments =
segDetectionResponse.segments();
                for (SegmentDetection detectedSegment : detectedSegments) {
                    String type = detectedSegment.type().toString();
                    if (type.contains(SegmentType.TECHNICAL_CUE.toString())) {
                        System.out.println("Technical Cue");
                        TechnicalCueSegment segmentCue =
detectedSegment.technicalCueSegment();
                        System.out.println("\tType: " + segmentCue.type());
                        System.out.println("\tConfidence: " +
segmentCue.confidence().toString());
                    }

                    if (type.contains(SegmentType.SHOT.toString())) {
                        System.out.println("Shot");
                        ShotSegment segmentShot = detectedSegment.shotSegment();
                        System.out.println("\tIndex " + segmentShot.index());
                        System.out.println("\tConfidence: " +
segmentShot.confidence().toString());
                    }

                    long seconds = detectedSegment.durationMillis();
                    System.out.println("\tDuration : " + seconds + "
milliseconds");
                    System.out.println("\tStart time code: " +
detectedSegment.startTimecodeSMPTE());
                    System.out.println("\tEnd time code: " +
detectedSegment.endTimecodeSMPTE());
                    System.out.println("\tDuration time code: " +
detectedSegment.durationSMPTE());
                    System.out.println();
                }

            } while (segDetectionResponse != null &&
segDetectionResponse.nextToken() != null);

        } catch (RekognitionException | InterruptedException e) {
```

```
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

Detecte texto en un vídeo almacenado en un bucket de Amazon S3.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.S3Object;
import software.amazon.awssdk.services.rekognition.model.NotificationChannel;
import software.amazon.awssdk.services.rekognition.model.Video;
import
 software.amazon.awssdk.services.rekognition.model.StartTextDetectionRequest;
import
 software.amazon.awssdk.services.rekognition.model.StartTextDetectionResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import
 software.amazon.awssdk.services.rekognition.model.GetTextDetectionResponse;
import software.amazon.awssdk.services.rekognition.model.GetTextDetectionRequest;
import software.amazon.awssdk.services.rekognition.model.VideoMetadata;
import software.amazon.awssdk.services.rekognition.model.TextDetectionResult;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class VideoDetectText {
    private static String startJobId = "";

    public static void main(String[] args) {
        final String usage = """

                Usage:    <bucket> <video> <topicArn> <roleArn>
```

```
                Where:
                   bucket - The name of the bucket in which the video is located
(for example, (for example, myBucket).\s
                   video - The name of video (for example, people.mp4).\s
                   topicArn - The ARN of the Amazon Simple Notification Service
(Amazon SNS) topic.\s
                   roleArn - The ARN of the AWS Identity and Access Management
(IAM) role to use.\s
                """;

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

       String bucket = args[0];
       String video = args[1];
       String topicArn = args[2];
       String roleArn = args[3];

       Region region = Region.US_EAST_1;
       RekognitionClient rekClient = RekognitionClient.builder()
               .region(region)
               .build();

       NotificationChannel channel = NotificationChannel.builder()
               .snsTopicArn(topicArn)
               .roleArn(roleArn)
               .build();

       startTextLabels(rekClient, channel, bucket, video);
       getTextResults(rekClient);
       System.out.println("This example is done!");
       rekClient.close();
   }

   public static void startTextLabels(RekognitionClient rekClient,
           NotificationChannel channel,
           String bucket,
           String video) {
       try {
           S3Object s3Obj = S3Object.builder()
                   .bucket(bucket)
                   .name(video)
```

```
                    .build();

            Video vidOb = Video.builder()
                    .s3Object(s3Obj)
                    .build();

            StartTextDetectionRequest labelDetectionRequest =
StartTextDetectionRequest.builder()
                    .jobTag("DetectingLabels")
                    .notificationChannel(channel)
                    .video(vidOb)
                    .build();

            StartTextDetectionResponse labelDetectionResponse =
rekClient.startTextDetection(labelDetectionRequest);
            startJobId = labelDetectionResponse.jobId();

        } catch (RekognitionException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void getTextResults(RekognitionClient rekClient) {
        try {
            String paginationToken = null;
            GetTextDetectionResponse textDetectionResponse = null;
            boolean finished = false;
            String status;
            int yy = 0;

            do {
                if (textDetectionResponse != null)
                    paginationToken = textDetectionResponse.nextToken();

                GetTextDetectionRequest recognitionRequest =
GetTextDetectionRequest.builder()
                        .jobId(startJobId)
                        .nextToken(paginationToken)
                        .maxResults(10)
                        .build();

                // Wait until the job succeeds.
                while (!finished) {
```

```
                    textDetectionResponse =
rekClient.getTextDetection(recognitionRequest);
                    status = textDetectionResponse.jobStatusAsString();

                    if (status.compareTo("SUCCEEDED") == 0)
                        finished = true;
                    else {
                        System.out.println(yy + " status is: " + status);
                        Thread.sleep(1000);
                    }
                    yy++;
                }

                finished = false;

                // Proceed when the job is done - otherwise VideoMetadata is
null.
                VideoMetadata videoMetaData =
textDetectionResponse.videoMetadata();
                System.out.println("Format: " + videoMetaData.format());
                System.out.println("Codec: " + videoMetaData.codec());
                System.out.println("Duration: " +
videoMetaData.durationMillis());
                System.out.println("FrameRate: " + videoMetaData.frameRate());
                System.out.println("Job");

                List<TextDetectionResult> labels =
textDetectionResponse.textDetections();
                for (TextDetectionResult detectedText : labels) {
                    System.out.println("Confidence: " +
detectedText.textDetection().confidence().toString());
                    System.out.println("Id : " +
detectedText.textDetection().id());
                    System.out.println("Parent Id: " +
detectedText.textDetection().parentId());
                    System.out.println("Type: " +
detectedText.textDetection().type());
                    System.out.println("Text: " +
detectedText.textDetection().detectedText());
                    System.out.println();
                }

            } while (textDetectionResponse != null &&
textDetectionResponse.nextToken() != null);
```

```
        } catch (RekognitionException | InterruptedException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

Detecte personas en un vídeo almacenado en un bucket de Amazon S3.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.S3Object;
import software.amazon.awssdk.services.rekognition.model.NotificationChannel;
import
 software.amazon.awssdk.services.rekognition.model.StartPersonTrackingRequest;
import software.amazon.awssdk.services.rekognition.model.Video;
import
 software.amazon.awssdk.services.rekognition.model.StartPersonTrackingResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import
 software.amazon.awssdk.services.rekognition.model.GetPersonTrackingResponse;
import
 software.amazon.awssdk.services.rekognition.model.GetPersonTrackingRequest;
import software.amazon.awssdk.services.rekognition.model.VideoMetadata;
import software.amazon.awssdk.services.rekognition.model.PersonDetection;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class VideoPersonDetection {
    private static String startJobId = "";

    public static void main(String[] args) {
```

```
        final String usage = """

                Usage:    <bucket> <video> <topicArn> <roleArn>

                Where:
                   bucket - The name of the bucket in which the video is located
(for example, (for example, myBucket).\s
                   video - The name of video (for example, people.mp4).\s
                   topicArn - The ARN of the Amazon Simple Notification Service
(Amazon SNS) topic.\s
                   roleArn - The ARN of the AWS Identity and Access Management
(IAM) role to use.\s
                """;

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

        String bucket = args[0];
        String video = args[1];
        String topicArn = args[2];
        String roleArn = args[3];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        NotificationChannel channel = NotificationChannel.builder()
                .snsTopicArn(topicArn)
                .roleArn(roleArn)
                .build();

        startPersonLabels(rekClient, channel, bucket, video);
        getPersonDetectionResults(rekClient);
        System.out.println("This example is done!");
        rekClient.close();
    }

    public static void startPersonLabels(RekognitionClient rekClient,
            NotificationChannel channel,
            String bucket,
            String video) {
        try {
```

```
            S3Object s3Obj = S3Object.builder()
                    .bucket(bucket)
                    .name(video)
                    .build();

            Video vidOb = Video.builder()
                    .s3Object(s3Obj)
                    .build();

            StartPersonTrackingRequest personTrackingRequest =
StartPersonTrackingRequest.builder()
                    .jobTag("DetectingLabels")
                    .video(vidOb)
                    .notificationChannel(channel)
                    .build();

            StartPersonTrackingResponse labelDetectionResponse =
rekClient.startPersonTracking(personTrackingRequest);
            startJobId = labelDetectionResponse.jobId();

        } catch (RekognitionException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void getPersonDetectionResults(RekognitionClient rekClient) {
        try {
            String paginationToken = null;
            GetPersonTrackingResponse personTrackingResult = null;
            boolean finished = false;
            String status;
            int yy = 0;

            do {
                if (personTrackingResult != null)
                    paginationToken = personTrackingResult.nextToken();

                GetPersonTrackingRequest recognitionRequest =
GetPersonTrackingRequest.builder()
                        .jobId(startJobId)
                        .nextToken(paginationToken)
                        .maxResults(10)
                        .build();
```

```
                // Wait until the job succeeds
                while (!finished) {

                    personTrackingResult =
rekClient.getPersonTracking(recognitionRequest);
                    status = personTrackingResult.jobStatusAsString();

                    if (status.compareTo("SUCCEEDED") == 0)
                        finished = true;
                    else {
                        System.out.println(yy + " status is: " + status);
                        Thread.sleep(1000);
                    }
                    yy++;
                }

                finished = false;

                // Proceed when the job is done - otherwise VideoMetadata is
null.
                VideoMetadata videoMetaData =
personTrackingResult.videoMetadata();

                System.out.println("Format: " + videoMetaData.format());
                System.out.println("Codec: " + videoMetaData.codec());
                System.out.println("Duration: " +
videoMetaData.durationMillis());
                System.out.println("FrameRate: " + videoMetaData.frameRate());
                System.out.println("Job");

                List<PersonDetection> detectedPersons =
personTrackingResult.persons();
                for (PersonDetection detectedPerson : detectedPersons) {
                    long seconds = detectedPerson.timestamp() / 1000;
                    System.out.print("Sec: " + seconds + " ");
                    System.out.println("Person Identifier: " +
detectedPerson.person().index());
                    System.out.println();
                }

            } while (personTrackingResult != null &&
personTrackingResult.nextToken() != null);
```

```
        } catch (RekognitionException | InterruptedException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- Para obtener detalles sobre la API, consulte los siguientes temas en la Referencia de la API de AWS SDK for Java 2.x .
  - GetCelebrityRecognition
  - GetContentModeration
  - GetLabelDetection
  - GetPersonTracking
  - GetSegmentDetection
  - GetTextDetection
  - StartCelebrityRecognition
  - StartContentModeration
  - StartLabelDetection
  - StartPersonTracking
  - StartSegmentDetection
  - StartTextDetection

Kotlin

SDK para Kotlin

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

Detecte rostros en un vídeo almacenado en un bucket de Amazon S3.

```
suspend fun startFaceDetection(
```

```
    channelVal: NotificationChannel?,
    bucketVal: String,
    videoVal: String,
) {
    val s3Obj =
        S3Object {
            bucket = bucketVal
            name = videoVal
        }
    val vidOb =
        Video {
            s3Object = s3Obj
        }

    val request =
        StartFaceDetectionRequest {
            jobTag = "Faces"
            faceAttributes = FaceAttributes.All
            notificationChannel = channelVal
            video = vidOb
        }

    RekognitionClient.fromEnvironment { region = "us-east-1" }.use { rekClient ->
        val startLabelDetectionResult = rekClient.startFaceDetection(request)
        startJobId = startLabelDetectionResult.jobId.toString()
    }
}

suspend fun getFaceResults() {
    var finished = false
    var status: String
    var yy = 0
    RekognitionClient.fromEnvironment { region = "us-east-1" }.use { rekClient ->
        var response: GetFaceDetectionResponse? = null

        val recognitionRequest =
            GetFaceDetectionRequest {
                jobId = startJobId
                maxResults = 10
            }

        // Wait until the job succeeds.
        while (!finished) {
            response = rekClient.getFaceDetection(recognitionRequest)
```

```
            status = response.jobStatus.toString()
            if (status.compareTo("Succeeded") == 0) {
                finished = true
            } else {
                println("$yy status is: $status")
                delay(1000)
            }
            yy++
        }

        // Proceed when the job is done - otherwise VideoMetadata is null.
        val videoMetaData = response?.videoMetadata
        println("Format: ${videoMetaData?.format}")
        println("Codec: ${videoMetaData?.codec}")
        println("Duration: ${videoMetaData?.durationMillis}")
        println("FrameRate: ${videoMetaData?.frameRate}")

        // Show face information.
        response?.faces?.forEach { face ->
            println("Age: ${face.face?.ageRange}")
            println("Face: ${face.face?.beard}")
            println("Eye glasses: ${face?.face?.eyeglasses}")
            println("Mustache: ${face.face?.mustache}")
            println("Smile: ${face.face?.smile}")
        }
    }
}
```

Detecte contenido inapropiado u ofensivo en un vídeo almacenado en un bucket de Amazon S3.

```
suspend fun startModerationDetection(
    channel: NotificationChannel?,
    bucketVal: String?,
    videoVal: String?,
) {
    val s3Obj =
        S3Object {
            bucket = bucketVal
            name = videoVal
        }
    val vidOb =
```

```
        Video {
            s3Object = s3Obj
        }
    val request =
        StartContentModerationRequest {
            jobTag = "Moderation"
            notificationChannel = channel
            video = vidOb
        }

    RekognitionClient.fromEnvironment { region = "us-east-1" }.use { rekClient ->
        val startModDetectionResult = rekClient.startContentModeration(request)
        startJobId = startModDetectionResult.jobId.toString()
    }
}

suspend fun getModResults() {
    var finished = false
    var status: String
    var yy = 0
    RekognitionClient { region = "us-east-1" }.use { rekClient ->
        var modDetectionResponse: GetContentModerationResponse? = null

        val modRequest =
            GetContentModerationRequest {
                jobId = startJobId
                maxResults = 10
            }

        // Wait until the job succeeds.
        while (!finished) {
            modDetectionResponse = rekClient.getContentModeration(modRequest)
            status = modDetectionResponse.jobStatus.toString()
            if (status.compareTo("Succeeded") == 0) {
                finished = true
            } else {
                println("$yy status is: $status")
                delay(1000)
            }
            yy++
        }

        // Proceed when the job is done - otherwise VideoMetadata is null.
        val videoMetaData = modDetectionResponse?.videoMetadata
```

```
        println("Format: ${videoMetaData?.format}")
        println("Codec: ${videoMetaData?.codec}")
        println("Duration: ${videoMetaData?.durationMillis}")
        println("FrameRate: ${videoMetaData?.frameRate}")

        modDetectionResponse?.moderationLabels?.forEach { mod ->
            val seconds: Long = mod.timestamp / 1000
            print("Mod label: $seconds ")
            println(mod.moderationLabel)
        }
    }
}
```

- Para obtener información sobre la API, consulte los siguientes temas en la Referencia de la API de AWS SDK para Kotlin.
  - GetCelebrityRecognition
  - GetContentModeration
  - GetLabelDetection
  - GetPersonTracking
  - GetSegmentDetection
  - GetTextDetection
  - StartCelebrityRecognition
  - StartContentModeration
  - StartLabelDetection
  - StartPersonTracking
  - StartSegmentDetection
  - StartTextDetection

Para obtener una lista completa de guías para desarrolladores del AWS SDK y ejemplos de código, consulteUso de Rekognition con un SDK AWS. En este tema también se incluye información sobre cómo comenzar a utilizar el SDK y detalles sobre sus versiones anteriores.

Detecte objetos en imágenes con Amazon Rekognition AWS mediante un SDK

Los siguientes ejemplos de código muestran cómo crear una aplicación que utilice Amazon Rekognition para detectar objetos por categoría en imágenes.

.NET

SDK para .NET

Muestra cómo utilizar la API de .NET de Amazon Rekognition para crear una aplicación que utilice Amazon Rekognition para identificar objetos por categoría en imágenes ubicadas en un bucket de Amazon Simple Storage Service (Amazon S3). La aplicación envía al administrador una notificación por correo electrónico con los resultados mediante Amazon Simple Email Service (Amazon SES).

Para ver el código fuente completo y las instrucciones sobre cómo configurarlo y ejecutarlo, consulta el ejemplo completo en GitHub.

Servicios utilizados en este ejemplo

- Amazon Rekognition
- Amazon S3
- Amazon SES

Java

SDK para Java 2.x

Muestra cómo utilizar la API de Java de Amazon Rekognition para crear una aplicación que utilice Amazon Rekognition para identificar objetos por categoría en imágenes ubicadas en un bucket de Amazon Simple Storage Service (Amazon S3). La aplicación envía al administrador una notificación por correo electrónico con los resultados mediante Amazon Simple Email Service (Amazon SES).

Para ver el código fuente completo y las instrucciones sobre cómo configurarlo y ejecutarlo, consulta el ejemplo completo en GitHub.

Servicios utilizados en este ejemplo

- Amazon Rekognition

- Amazon S3
- Amazon SES

JavaScript

SDK para JavaScript (v3)

Muestra cómo utilizar Amazon Rekognition AWS SDK para JavaScript con el para crear una aplicación que utilice Amazon Rekognition para identificar objetos por categoría en imágenes ubicadas en un bucket de Amazon Simple Storage Service (Amazon S3). La aplicación envía al administrador una notificación por correo electrónico con los resultados mediante Amazon Simple Email Service (Amazon SES).

Aprenda cómo:

- Crear un usuario no autenticado con Amazon Cognito.
- Analizar imágenes en busca de objetos con Amazon Rekognition.
- Verificar una dirección de correo electrónico de Amazon SES.
- Enviar una notificación por correo electrónico con Amazon SES.

Para obtener el código fuente completo y las instrucciones sobre cómo configurarlo y ejecutarlo, consulte el ejemplo completo en. GitHub

Servicios utilizados en este ejemplo

- Amazon Rekognition
- Amazon S3
- Amazon SES

Kotlin

SDK para Kotlin

Muestra cómo utilizar la API de Kotlin de Amazon Rekognition para crear una aplicación que utilice Amazon Rekognition para identificar objetos por categoría en imágenes ubicadas en un bucket de Amazon Simple Storage Service (Amazon S3). La aplicación envía al administrador una notificación por correo electrónico con los resultados mediante Amazon Simple Email Service (Amazon SES).

Para ver el código fuente completo y las instrucciones sobre cómo configurarlo y ejecutarlo, consulta el ejemplo completo en GitHub.

Servicios utilizados en este ejemplo

- Amazon Rekognition
- Amazon S3
- Amazon SES

Python

SDK para Python (Boto3)

Le muestra cómo utilizar el AWS SDK para Python (Boto3) para crear una aplicación web que le permita hacer lo siguiente:

- Subir fotos en un bucket de Amazon Simple Storage Service (Amazon S3).
- Utilizar Amazon Rekognition para analizar y etiquetar las fotos.
- Utilice Amazon Simple Email Service (Amazon SES) para enviar informes de análisis de imágenes por correo electrónico.

Este ejemplo contiene dos componentes principales: una página web escrita en JavaScript React y un servicio REST escrito en Python creado con Flask-RESTful.

Puede utilizar la página web de React para:

- Mostrar una lista de imágenes almacenadas en el bucket de S3.
- Subir imágenes desde la computadora en el bucket de S3.
- Mostrar imágenes y etiquetas que identifican los elementos detectados en la imagen.
- Obtener un informe de todas las imágenes del bucket de S3 y enviar un correo electrónico del informe.

La página web llama al servicio REST. El servicio envía solicitudes a AWS para llevar a cabo las siguientes acciones:

- Obtener y filtrar la lista de imágenes del bucket de S3.
- Subir fotos en el bucket de S3.
- Utilizar Amazon Rekognition para analizar fotos individuales y obtener una lista de etiquetas que identifican los elementos detectados en la foto.

- Analizar todas las fotos del bucket de S3 y usar Amazon SES para enviar un informe por correo electrónico.

Para ver el código fuente completo y las instrucciones sobre cómo configurarlo y ejecutarlo, consulta el ejemplo completo en GitHub.

Servicios utilizados en este ejemplo

- Amazon Rekognition
- Amazon S3
- Amazon SES

Para obtener una lista completa de guías para desarrolladores del AWS SDK y ejemplos de código, consulteUso de Rekognition con un SDK AWS. En este tema también se incluye información sobre cómo comenzar a utilizar el SDK y detalles sobre sus versiones anteriores.

Detecte personas y objetos en un vídeo con Amazon Rekognition AWS mediante un SDK

Los siguientes ejemplos de código indican cómo detectar personas y objetos en un video con Amazon Rekognition.

Java

SDK para Java 2.x

Muestra cómo utilizar la API Java de Amazon Rekognition para crear una aplicación que detecte rostros y objetos en vídeos ubicados en un bucket de Amazon Simple Storage Service (Amazon S3). La aplicación envía al administrador una notificación por correo electrónico con los resultados mediante Amazon Simple Email Service (Amazon SES).

Para ver el código fuente completo y las instrucciones sobre cómo configurarlo y ejecutarlo, consulta el ejemplo completo en GitHub.

Servicios utilizados en este ejemplo

- Amazon Rekognition
- Amazon S3
- Amazon SES

- Amazon SNS
- Amazon SQS

Python

SDK para Python (Boto3)

Utilice Amazon Rekognition para detectar caras, objetos y personas en videos iniciando trabajos de detección asíncronos. Este ejemplo también configura Amazon Rekognition para que notifique un tema de Amazon Simple Notification Service (Amazon SNS) cuando se finalicen los trabajos y suscriba una cola de Amazon Simple Queue Service (Amazon SQS) al tema. Cuando la cola recibe un mensaje sobre un trabajo, se recupera el trabajo y se muestran los resultados

Este ejemplo se ve mejor en GitHub. Para obtener el código fuente completo y las instrucciones sobre cómo configurarlo y ejecutarlo, consulte el ejemplo completo en GitHub.

Servicios utilizados en este ejemplo

- Amazon Rekognition
- Amazon S3
- Amazon SES
- Amazon SNS
- Amazon SQS

Para obtener una lista completa de guías para desarrolladores del AWS SDK y ejemplos de código, consulteUso de Rekognition con un SDK AWS. En este tema también se incluye información sobre cómo comenzar a utilizar el SDK y detalles sobre sus versiones anteriores.

Guarde el EXIF y otra información de imagen mediante un SDK AWS

En el siguiente ejemplo de código, se muestra cómo:

- Obtener información EXIF de un archivo JPG, JPEG o PNG.
- Subir el archivo de imagen en un bucket de Amazon S3.
- Usar Amazon Rekognition para identificar los tres atributos principales (etiquetas) en el archivo.
- Agregar la información EXIF y de etiquetas a una tabla de Amazon DynamoDB de la región.

Rust

SDK para Rust

Obtenga información EXIF de un archivo JPG, JPEG o PNG, cargue el archivo de imagen en un bucket de Amazon S3, utilice Amazon Rekognition para identificar los tres atributos principales (etiquetas de Amazon Rekognition) en el archivo y añada la información EXIF y de etiquetas a una tabla de Amazon DynamoDB de la región.

Para ver el código fuente completo y las instrucciones sobre cómo configurarlo y ejecutarlo, consulta el ejemplo completo en GitHub.

Servicios utilizados en este ejemplo

- DynamoDB
- Amazon Rekognition
- Amazon S3

Para obtener una lista completa de guías para desarrolladores del AWS SDK y ejemplos de código, consulteUso de Rekognition con un SDK AWS. En este tema también se incluye información sobre cómo comenzar a utilizar el SDK y detalles sobre sus versiones anteriores.

referencia de la API

La referencia de la API de Amazon Rekognition ahora está en la Referencia de API de Amazon Rekognition.

Seguridad de Amazon Rekognition

La seguridad en la nube AWS es la máxima prioridad. Como AWS cliente, usted se beneficia de un centro de datos y una arquitectura de red diseñados para cumplir con los requisitos de las organizaciones más sensibles a la seguridad.

Utilice los siguientes temas para aprender a proteger los recursos de Amazon Rekognition.

Temas

- Administración de identidades y accesos para Amazon Rekognition
- Protección de datos en Amazon Rekognition
- Uso de Amazon Rekognition con puntos de conexión de Amazon VPC
- Validación de la conformidad para Amazon Rekognition
- Resiliencia en Amazon Rekognition
- Configuración y análisis de vulnerabilidades en Amazon Rekognition
- Prevención de la sustitución confusa entre servicios
- Seguridad de la infraestructura de Amazon Rekognition

Administración de identidades y accesos para Amazon Rekognition

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 se puede autenticar (iniciar sesión) y autorizar (tener permisos) para utilizar los recursos de Amazon Rekognition. La IAM es una Servicio de AWS herramienta que puede utilizar sin coste adicional.

Temas

- Público
- Autenticación con identidades
- Administración del acceso con políticas
- Cómo funciona Amazon Rekognition con IAM
- AWS políticas gestionadas para Amazon Rekognition
- Prácticas recomendadas sobre las políticas

- Uso de la consola de Amazon Rekognition
- Ejemplos de uso de políticas de Amazon Rekognition basadas en identidades
- Ejemplos de políticas de Amazon Rekognition basadas en recursos
- Solución de problemas de identidad y acceso de Amazon Rekognition

Público

La forma de usar AWS Identity and Access Management (IAM) varía según la función que desempeñes:

- Usuario del servicio: solicite permisos al administrador si no puede acceder a las características (consulte Solución de problemas de identidad y acceso de Amazon Rekognition).
- Administrador del servicio: determine el acceso de los usuarios y envíe las solicitudes de permiso (consulte Cómo funciona Amazon Rekognition con IAM).
- Administrador de IAM: escribe las políticas para administrar el acceso (consulte Ejemplos de uso de políticas de Amazon Rekognition basadas en identidades).

Autenticación con identidades

La autenticación es la forma en que inicias sesión AWS con tus credenciales de identidad. Debe autenticarse como usuario de Usuario raíz de la cuenta de AWS IAM o asumir una función de IAM.

Puede iniciar sesión como una identidad federada con las credenciales de una fuente de identidad, como AWS IAM Identity Center (IAM Identity Center), la autenticación de inicio de sesión único o las credenciales. Google/Facebook Para obtener más información sobre el inicio de sesión, consulte Cómo iniciar sesión en la Cuenta de AWS en la Guía del usuario de AWS Sign-In .

Para el acceso programático, AWS proporciona un SDK y una CLI para firmar criptográficamente las solicitudes. Para obtener más información, consulte AWS Signature Version 4 para solicitudes de API en la Guía del usuario de IAM.

Usuarios y grupos de IAM

Un usuario de IAM es una identidad con permisos específicos para una sola persona o aplicación. Recomendamos el uso de credenciales temporales en lugar de usuarios de IAM con credenciales de larga duración. Para obtener más información, consulte Exigir a los usuarios humanos que utilicen la

federación con un proveedor de identidad para acceder AWS mediante credenciales temporales en la Guía del usuario de IAM.

Un grupo de IAM especifica un conjunto de usuarios de IAM y facilita la administración de los permisos para grupos grandes de usuarios. 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 con permisos específicos que proporciona credenciales temporales. Puede asumir un rol cambiando de un rol de usuario a uno de IAM (consola) o llamando a una AWS CLI operación de AWS API. Para obtener más información, consulte Métodos para asumir un rol en la Guía del usuario de IAM.

Los roles de IAM son útiles para el acceso de usuario federado, los permisos de usuario de IAM temporales, el acceso entre cuentas, el acceso entre servicios y las aplicaciones que se ejecutan en Amazon EC2. Para obtener más información, consulte Acceso a recursos entre cuentas en IAM en la Guía del usuario de IAM.

Administración del acceso con políticas

El acceso se controla AWS creando políticas y adjuntándolas a AWS identidades o recursos. Una política define los permisos cuando están asociados a una identidad o un recurso. AWS evalúa estas políticas cuando un director hace una solicitud. La mayoría de las políticas se almacenan AWS como documentos JSON. Para obtener más información sobre los documentos de políticas de JSON, consulte Información general de políticas de JSON en la Guía del usuario de IAM.

Mediante las políticas, los administradores especifican quién tiene acceso a qué, definiendo qué entidad principal puede realizar acciones sobre qué recursos y en qué condiciones.

De forma predeterminada, los usuarios y los roles no tienen permisos. Un administrador de IAM crea políticas de IAM y las agrega a roles, que los usuarios pueden asumir posteriormente. Las políticas de IAM definen permisos independientemente del método que se utilice para realizar la operación.

Uso de políticas basadas en identidades

Las políticas basadas en identidad son documentos de política de permisos JSON que asocia a una identidad (usuario, grupo o rol). Estas políticas controlan qué acciones pueden realizar las identidades, en qué recursos y en qué condiciones. Para obtener más información sobre cómo crear

una política basada en la identidad, consulte Definición de permisos de IAM personalizados con políticas administradas por el cliente en la Guía del usuario de IAM.

Las políticas basadas en identidad pueden ser políticas insertadas (incrustadas directamente en una sola identidad) o políticas administradas (políticas independientes asociadas a varias identidades). Para obtener información sobre cómo elegir entre políticas administradas e insertadas, consulte Selección entre políticas administradas y políticas insertadas en la Guía del usuario de IAM.

Uso de políticas basadas en recursos

Las políticas basadas en recursos son documentos de política JSON que se asocian a un recurso. Los ejemplos incluyen 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 pueden utilizarlos para controlar el acceso a un recurso específico. Debe especificar una entidad principal en una política basada en recursos.

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 que pueden establecer los permisos máximos otorgados por los tipos de políticas más comunes:

- Límites de permisos: establecen los permisos máximos que una política basada en identidad puede conceder a una entidad de IAM. Para obtener más información, consulte Límites de permisos para las entidades de IAM en la Guía del usuario de IAM.
- Políticas de control de servicios (SCPs): especifican los permisos máximos para una organización o unidad organizativa en AWS Organizations. Para obtener más información, consulte Políticas de control de servicios en la Guía del usuario de AWS Organizations .

- Políticas de control de recursos (RCPs): establece los permisos máximos disponibles para los recursos de tus cuentas. Para obtener más información, consulte Políticas de control de recursos (RCPs) en la Guía del AWS Organizations usuario.
- Políticas de sesión: políticas avanzadas que se pasan como parámetro cuando se crea una sesión temporal para un rol o un usuario federado. Para obtener más información, consulte 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 Rekognition con IAM

Antes de utilizar IAM para administrar el acceso a Amazon Rekognition, debe conocer qué características de IAM están disponibles con Amazon Rekognition. Para obtener una visión general de cómo Amazon Rekognition AWS y otros servicios funcionan con IAM AWS , consulte Servicios que funcionan con IAM en la Guía del usuario de IAM.

Temas

- Políticas de Amazon Rekognition basadas en identidades
- Políticas basadas en recursos de Amazon Rekognition
- Roles de IAM de Amazon Rekognition

Políticas de Amazon Rekognition basadas en identidades

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. Amazon Rekognition admite acciones, claves de condiciones y recursos específicos. Para obtener información sobre todos los elementos que utiliza en una política JSON, consulte Referencia de los elementos de las políticas JSON de IAM en la Guía del usuario de IAM.

Acciones

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.

El elemento `Action` de una política JSON describe las acciones que puede utilizar para conceder o denegar el acceso en una política. Incluya acciones en una política para conceder permisos y así llevar a cabo la operación asociada.

Las acciones de políticas de Amazon Rekognition utilizan el siguiente prefijo antes de la acción: `rekognition:`. Por ejemplo, para conceder permiso a alguien para detectar objetos, escenas o conceptos en una imagen con la operación `DetectLabels` de la API de Amazon Rekognition, debe incluir la acción `rekognition:DetectLabels` en su política. Las instrucciones de la política deben incluir un elemento `Action` o un elemento `NotAction`. Amazon Rekognition define su propio conjunto de acciones que describen las tareas que se pueden realizar con este servicio.

Para especificar varias acciones en una única instrucción, sepárelas con comas del siguiente modo:

```
"Action": [
      "rekognition:action1",
      "rekognition:action2"
```

Puede utilizar caracteres comodín para especificar varias acciones (*). Por ejemplo, para especificar todas las acciones que comiencen con la palabra `Describe`, incluya la siguiente acción:

```
"Action": "rekognition:Describe*"
```

Para ver una lista de las acciones de Amazon Rekognition, consulte Acciones definidas por Amazon Rekognition en la Guía del usuario de IAM.

Recursos

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.

El elemento `Resource` de la política JSON especifica el objeto u objetos a los que se aplica la acción. Como práctica recomendada, especifique un recurso utilizando el Nombre de recurso de Amazon (ARN). En el caso de las acciones que no admiten permisos por recurso, utilice un carácter comodín (*) para indicar que la instrucción se aplica a todos los recursos.

```
"Resource": "*"
```

Para obtener más información sobre el formato de ARNs, consulte Amazon Resource Names (ARNs) y AWS Service Namespaces.

Por ejemplo, para especificar la colección `MyCollection` en la instrucción, utilice el siguiente ARN:

```
"Resource": "arn:aws:rekognition:us-east-1:123456789012:collection/MyCollection"
```

Para especificar todas las instancias que pertenecen a una cuenta específica, utilice el carácter comodín (*):

```
"Resource": "arn:aws:rekognition:us-east-1:123456789012:collection/*"
```

Algunas acciones de Amazon Rekognition, como las que se utilizan para crear recursos, no se pueden llevar a cabo en un recurso específico. En dichos casos, debe utilizar el carácter comodín (*).

```
"Resource": "*"
```

Para ver una lista de los tipos de recursos de Amazon Rekognition ARNs y sus tipos de recursos, consulte Recursos definidos por Amazon Rekognition en la Guía del usuario de IAM. Para obtener información acerca de las acciones con las que puede especificar el ARN de cada recurso, consulte Acciones definidas por Amazon Rekognition.

Claves de condición

Amazon Rekognition no proporciona ninguna clave de condición específica del servicio, pero sí admite el uso de algunas claves de condición globales. 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.

Políticas basadas en recursos de Amazon Rekognition

Políticas basadas en recursos de Amazon Rekognition para operaciones de copia de modelos de Etiquetas personalizadas. Para obtener más información, vea Amazon Rekognition resource-based policy examples.

Otros servicios, como Amazon S3, también admiten políticas de permisos basadas en recursos. Por ejemplo, puede asociar una política a un bucket de S3 para administrar los permisos de acceso a dicho bucket.

Para tener acceso a las imágenes almacenadas en un bucket de Amazon S3, debe tener permiso de acceso al objeto en el bucket de S3. Con este permiso, Amazon Rekognition puede descargar imágenes del bucket de S3. La siguiente política de ejemplo permite al usuario realizar la acción `s3:GetObject` en el bucket de S3 denominado amzn-s3-demo-bucket3.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "s3:GetObject",
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket3/*"
            ]
        }
    ]
}
```

Para usar un bucket de S3 con el control de versiones habilitado, añada la acción `s3:GetObjectVersion`, como se muestra en el siguiente ejemplo.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket3/*"
            ]
        }
```

```
    ]
}
```

Roles de IAM de Amazon Rekognition

Un rol de IAM es una entidad de tu AWS cuenta que tiene permisos específicos.

Uso de credenciales temporales con Amazon Rekognition

Puede utilizar credenciales temporales para iniciar sesión con federación, asumir un rol de IAM o asumir un rol de acceso entre cuentas. Para obtener credenciales de seguridad temporales, puede llamar a operaciones de AWS STS API como AssumeRoleo GetFederationToken.

Amazon Rekognition admite el uso de credenciales temporales.

Roles vinculados a servicios

Los roles vinculados a un servicio permiten a AWS los servicios acceder a los recursos de otros servicios para completar una acción en tu nombre. Los roles vinculados a servicios aparecen en la cuenta de IAM y son propiedad del servicio. Un administrador de IAM puede ver, pero no editar, los permisos de los roles vinculados a servicios.

Amazon Rekognition no admite roles vinculados a servicios.

Roles de servicio

Esta característica permite que un servicio asuma un rol de servicio en su nombre. Este rol permite que el servicio obtenga acceso a los recursos de otros servicios para completar una acción en su nombre. Los roles de servicio aparecen en su cuenta de IAM y son propiedad de la cuenta. Esto significa que un administrador de IAM puede cambiar los permisos de este rol. Sin embargo, hacerlo podría deteriorar la funcionalidad del servicio.

Amazon Rekognition admite roles de servicio.

El uso de un rol de servicio puede crear un problema de seguridad en el que Amazon Rekognition se utilice para llamar a otro servicio y actuar sobre recursos a los que no debería tener acceso. Para mantener la seguridad de su cuenta, debe limitar el alcance del acceso de Amazon Rekognition únicamente a los recursos que esté utilizando. Para ello, puede adjuntar una política de confianza a su rol de servicio de IAM. Para obtener información sobre cómo hacerlo, consulte Prevención de la sustitución confusa entre servicios.

Elegir un rol de IAM en Amazon Rekognition

Al configurar Amazon Rekognition para analizar vídeos almacenados, debe elegir un rol para permitir a Amazon Rekognition acceder a Amazon SNS en su nombre. Si ha creado previamente un rol de servicio o un rol vinculado a servicios, Amazon Rekognition le proporciona una lista de roles para elegir. Para obtener más información, consulte the section called “Configuración de Amazon Rekognition Video”.

Ejemplos: configuración de Amazon Rekognition para acceder a imágenes almacenadas en un bucket de Amazon S3

A continuación, se muestra un ejemplo de cómo puede configurar Amazon Rekognition para analizar imágenes almacenadas en un bucket de Amazon S3. Si prefiere utilizar Amazon Rekognition para analizar imágenes almacenadas en un bucket de Amazon S3, haga lo siguiente:

1. Asegúrese de que su IAM user/role (el cliente) tenga permiso para llamar a las operaciones de la API Amazon Rekognition pertinentes (por ejemplo, etc.) DetectLabels DetectFaces

   Asocie una política basada en identidad que otorgue los permisos adecuados para invocar las operaciones de API que desee. Por ejemplo, para conceder a su rol permisos para llamar a `DetectLabels` y `DetectFaces`, asociará a su rol una política similar a la siguiente:

   JSON

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

2. El servicio de Amazon Rekognition necesita permiso para acceder a su bucket de Amazon S3. Cree un rol de servicio de IAM, que deberá transferir a Amazon Rekognition cuando realice

llamadas a la API. Asegúrese de que el rol de servicio: Confía en la entidad principal de Amazon Rekognition, tenga permisos `s3:GetObject` para su bucket.

La política de confianza podría tener el siguiente aspecto:

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "rekognition.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

La política basada en identidad asociada al rol de servicio podría tener este aspecto:

JSON

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

AWS políticas gestionadas para Amazon Rekognition

Para añadir permisos a usuarios, grupos y roles, es más fácil usar políticas AWS administradas que escribirlas usted mismo. Se necesita tiempo y experiencia para crear políticas administradas por el cliente de IAM que proporcionen a su equipo solo los permisos necesarios. Para empezar rápidamente, puedes usar nuestras políticas AWS gestionadas. Estas políticas cubren casos de uso comunes y están disponibles en tu AWS cuenta. Para obtener más información sobre las políticas AWS administradas, consulte las políticas AWS administradas en la Guía del usuario de IAM.

AWS los servicios mantienen y AWS actualizan las políticas gestionadas. No puede cambiar los permisos en las políticas AWS gestionadas. En ocasiones, los servicios agregan permisos adicionales a una política administrada de AWS para admitir características nuevas. Este tipo de actualización afecta a todas las identidades (usuarios, grupos y roles) donde se asocia la política. Es más probable que los servicios actualicen una política administrada de AWS cuando se lanza una nueva característica o cuando se ponen a disposición nuevas operaciones. Los servicios no eliminan los permisos de una política AWS administrada, por lo que las actualizaciones de la política no afectarán a los permisos existentes.

Además, AWS admite políticas administradas para funciones laborales que abarcan varios servicios. Por ejemplo, la política ReadOnlyAccess AWS gestionada proporciona acceso de solo lectura a todos los AWS servicios y recursos. Cuando un servicio lanza una nueva función, AWS agrega permisos de solo lectura para nuevas operaciones y recursos. Para obtener una lista y descripciones de las políticas de funciones de trabajo, consulte Políticas administradas de AWS para funciones de trabajo en la Guía del usuario de IAM.

Política gestionada por AWS: AmazonRekognitionFullAccess

`AmazonRekognitionFullAccess` concede acceso completo a los recursos de Amazon Rekognition, incluida la creación y eliminación de colecciones.

Puede adjuntar la política de `AmazonRekognitionFullAccess` a las identidades de IAM.

Detalles de los permisos

Esta política incluye los siguientes permisos.

JSON

```
{
```

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

Política gestionada por AWS: AmazonRekognitionReadOnlyAccess

`AmazonRekognitionReadOnlyAccess` concede acceso de solo lectura a recursos de Amazon Rekognition.

Puede adjuntar la política de `AmazonRekognitionReadOnlyAccess` a las identidades de IAM.

Detalles de los permisos

Esta política incluye los siguientes permisos.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AmazonRekognitionReadOnlyAccess",
            "Effect": "Allow",
            "Action": [
                "rekognition:CompareFaces",
                "rekognition:DetectFaces",
                "rekognition:DetectLabels",
                "rekognition:ListCollections",
                "rekognition:ListFaces",
                "rekognition:SearchFaces",
                "rekognition:SearchFacesByImage",
                "rekognition:DetectText",
                "rekognition:GetCelebrityInfo",
                "rekognition:RecognizeCelebrities",
```

```
                "rekognition:DetectModerationLabels",
                "rekognition:GetLabelDetection",
                "rekognition:GetFaceDetection",
                "rekognition:GetContentModeration",
                "rekognition:GetPersonTracking",
                "rekognition:GetCelebrityRecognition",
                "rekognition:GetFaceSearch",
                "rekognition:GetTextDetection",
                "rekognition:GetSegmentDetection",
                "rekognition:DescribeStreamProcessor",
                "rekognition:ListStreamProcessors",
                "rekognition:DescribeProjects",
                "rekognition:DescribeProjectVersions",
                "rekognition:DetectCustomLabels",
                "rekognition:DetectProtectiveEquipment",
                "rekognition:ListTagsForResource",
                "rekognition:ListDatasetEntries",
                "rekognition:ListDatasetLabels",
                "rekognition:DescribeDataset",
                "rekognition:ListProjectPolicies",
                "rekognition:ListUsers",
                "rekognition:SearchUsers",
                "rekognition:SearchUsersByImage",
                "rekognition:GetMediaAnalysisJob",
                "rekognition:ListMediaAnalysisJobs"
            ],
            "Resource": "*"
        }
    ]
}
```

Política gestionada por AWS: AmazonRekognitionServiceRole

`AmazonRekognitionServiceRole` permite a Amazon Rekognition llamar a los servicios de Amazon Kinesis Data Streams y Amazon SNS en su nombre.

Puede asociar la política `AmazonRekognitionServiceRole` a las identidades de IAM.

Si utiliza este rol de servicio, debe proteger su cuenta limitando el alcance del acceso de Amazon Rekognition únicamente a los recursos que utilice. Para ello, puede adjuntar una política de confianza a su rol de servicio de IAM. Para obtener información sobre cómo hacerlo, consulte Prevención de la sustitución confusa entre servicios.

Detalles de los permisos

Esta política incluye los siguientes permisos.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "sns:Publish"
            ],
            "Resource": "arn:aws:sns:*:*:AmazonRekognition*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "kinesis:PutRecord",
                "kinesis:PutRecords"
            ],
            "Resource": "arn:aws:kinesis:*:*:stream/AmazonRekognition*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "kinesisvideo:GetDataEndpoint",
                "kinesisvideo:GetMedia"
            ],
            "Resource": "*"
        }
    ]
}
```

Política gestionada por AWS: AmazonRekognitionCustomLabelsFullAccess

Esta política es para los usuarios de Etiquetas personalizadas de Amazon Rekognition. Utilice la AmazonRekognitionCustomLabelsFullAccess política para permitir a los usuarios un acceso total a la API de etiquetas personalizadas de Amazon Rekognition y a los buckets de consola creados por la consola de etiquetas personalizadas de Amazon Rekognition.

Detalles de los permisos

Esta política incluye los siguientes permisos.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:ListBucket",
                "s3:ListAllMyBuckets",
                "s3:GetBucketAcl",
                "s3:GetBucketLocation",
                "s3:GetObject",
                "s3:GetObjectAcl",
                "s3:GetObjectTagging",
                "s3:GetObjectVersion",
                "s3:PutObject"
            ],
            "Resource": "arn:aws:s3:::*custom-labels*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "rekognition:CopyProjectVersion",
                "rekognition:CreateProject",
                "rekognition:CreateProjectVersion",
                "rekognition:StartProjectVersion",
                "rekognition:StopProjectVersion",
                "rekognition:DescribeProjects",
                "rekognition:DescribeProjectVersions",
                "rekognition:DetectCustomLabels",
                "rekognition:DeleteProject",
                "rekognition:DeleteProjectVersion",
                "rekognition:TagResource",
                "rekognition:UntagResource",
                "rekognition:ListTagsForResource",
                "rekognition:CreateDataset",
                "rekognition:ListDatasetEntries",
                "rekognition:ListDatasetLabels",
```

```
                "rekognition:DescribeDataset",
                "rekognition:UpdateDatasetEntries",
                "rekognition:DistributeDatasetEntries",
                "rekognition:DeleteDataset",
                "rekognition:PutProjectPolicy",
                "rekognition:ListProjectPolicies",
                "rekognition:DeleteProjectPolicy"
            ],
            "Resource": "*"
        }
    ]
}
```

Amazon Rekognition AWS actualiza las políticas gestionadas

Consulte los detalles sobre las actualizaciones de las políticas AWS gestionadas de Amazon Rekognition 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 la API de Amazon Rekognition.

Cambio	Descripción	Fecha
Las acciones que implican trabajos de análisis de contenido multimedia se han agregado a la siguiente política administrada: • Política gestionada por AWS: AmazonRekognitionReadOnlyAccess	Amazon Rekognition ha añadido las siguiente s acciones a la política administrada de `AmazonRekognitionReadOnlyAccess` : • `GetMediaAnalysisJob` • `ListMediaAnalysisJob`	31 de octubre de 2023
Las acciones que implican la administración de usuarios se han agregado a la siguiente política administrada:	Amazon Rekognition ha añadido las siguiente s acciones a la política administrada de `AmazonRek`	12 de junio de 2023

Cambio	Descripción	Fecha
• Política gestionada por AWS: AmazonRekognitionReadOnlyAccess	ognitionReadOnlyAccess : • ListUsers • SearchUsers • SearchUsersByImage	
Se han agregado Actions for Model ProjectPolicy Copy y etiquetas personalizadas a las siguientes políticas administradas: • Política gestionada por AWS: AmazonRekognitionFullAccess • Política gestionada por AWS: AmazonRekognitionCustomLabelsFullAccess	Amazon Rekognition agregó las siguientes acciones a las políticas administradas AmazonRekognitionCustomLabelsFullAccess y AmazonRekognitionFullAccess : • CopyProjectVersion • PutProjectPolicy • ListProjectPolicies • DeleteProjectPolicy	21 de julio de 2022
Se han agregado acciones para etiquetas personalizadas Model ProjectPolicy Copy y se han agregado a las siguientes políticas administradas: • Política gestionada por AWS: AmazonRekognitionReadOnlyAccess	Amazon Rekognition ha añadido las siguientes acciones a la política administrada de AmazonRekognitionReadOnlyAccess : • ListProjectPolicies	21 de julio de 2022

Cambio	Descripción	Fecha
Actualización de la administración de conjuntos de datos para las siguientes políticas administradas: • Política gestionada por AWS: AmazonRekognitionReadOnlyAccess • Política gestionada por AWS: AmazonRekognitionFullAccess • Política gestionada por AWS: AmazonRekognitionCustomLabelsFullAccess	Amazon Rekognition agregó las siguientes acciones AmazonRekognitionReadOnlyAccess a AmazonRekognitionFullOnlyAccess las políticas, y administró: AmazonRekognitionCustomLabelsFullAccess • `CreateDataset` • `ListDatasetEntries` • `ListDatasetLabels` • `DescribeDataset` • `UpdateDatasetEntries` • `DistributeDatasetEntries` • `DeleteDataset`	1 de noviembre de 2021
Actualización de etiquetado para Política gestionada por AWS: AmazonRekognitionReadOnlyAccess y Política gestionada por AWS: AmazonRekognitionFullAccess	Amazon Rekognition agregó nuevas acciones de etiquetado a las políticas y. AmazonRekognitionFullAccess AmazonRekognitionReadOnlyAccess	2 de abril de 2021
Amazon Rekognition comenzó a hacer el seguimiento de los cambios	Amazon Rekognition comenzó a realizar un seguimiento de los cambios en sus políticas gestionadas. AWS	2 de abril de 2021

Prácticas recomendadas sobre las políticas

Las políticas basadas en identidades determinan si alguien puede crear, eliminar o acceder a los recursos de Amazon Rekognition 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 para 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, consulte 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 pueden 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, consulte Políticas y permisos en IAM en la Guía del usuario de IAM.
- Utilice condiciones en las políticas de IAM para restringir aún más el acceso: puede agregar una condición a sus políticas para limitar el acceso a las acciones y los recursos. Por ejemplo, puede 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 CloudFormation. Para obtener más información, consulte 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.

Uso de la consola de Amazon Rekognition

Con la excepción de la característica de Etiquetas personalizadas de Amazon Rekognition, Amazon Rekognition no requiere permisos adicionales al usar la consola de Amazon Rekognition. Para obtener información sobre las Etiquetas personalizadas de Amazon Rekognition, consulte Step 5: Set Up Amazon Rekognition Custom Labels Console Permissions.

No es necesario que concedas permisos mínimos de consola a los usuarios que solo realizan llamadas a la API o a la AWS CLI API. AWS En su lugar, permite acceso únicamente a las acciones que coincidan con la operación de API que intenta realizar.

Ejemplos de uso de políticas de Amazon Rekognition basadas en identidades

De forma predeterminada, los usuarios y roles no tienen permiso para crear ni modificar los recursos de Amazon Rekognition. Tampoco pueden realizar tareas con la AWS API Consola de administración de AWS AWS CLI, o. Un administrador de IAM debe crear políticas de IAM que concedan permisos a los usuarios y a los roles para realizar operaciones de la API concretas en los recursos especificados que necesiten. El administrador debe asociar esas políticas a los usuarios o grupos que necesiten esos permisos.

Para obtener información acerca de cómo crear una política basada en identidad de IAM con estos documentos de políticas JSON de ejemplo, consulte Creación de políticas en la pestaña JSON en la Guía del usuario de IAM.

Temas

- Ejemplo de políticas de Etiquetas personalizadas de Amazon Rekognition
- Ejemplo 1: Permitir al usuario acceso de solo lectura a los recursos
- Ejemplo 2: Permitir al usuario acceso completo a los recursos
- Cómo permitir a los usuarios consultar sus propios permisos

Ejemplo de políticas de Etiquetas personalizadas de Amazon Rekognition

Puede crear políticas basadas en la identidad para Etiquetas personalizadas de Amazon Rekognition. Para obtener más información, consulte Seguridad.

Ejemplo 1: Permitir al usuario acceso de solo lectura a los recursos

En el siguiente ejemplo se concede acceso de solo lectura a los recursos de Amazon Rekognition.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "rekognition:CompareFaces",
                "rekognition:DetectFaces",
                "rekognition:DetectLabels",
                "rekognition:ListCollections",
                "rekognition:ListFaces",
                "rekognition:SearchFaces",
                "rekognition:SearchFacesByImage",
                "rekognition:DetectText",
                "rekognition:GetCelebrityInfo",
                "rekognition:RecognizeCelebrities",
                "rekognition:DetectModerationLabels",
                "rekognition:GetLabelDetection",
                "rekognition:GetFaceDetection",
                "rekognition:GetContentModeration",
                "rekognition:GetPersonTracking",
                "rekognition:GetCelebrityRecognition",
                "rekognition:GetFaceSearch",
                "rekognition:GetTextDetection",
                "rekognition:GetSegmentDetection",
                "rekognition:DescribeStreamProcessor",
                "rekognition:ListStreamProcessors",
                "rekognition:DescribeProjects",
                "rekognition:DescribeProjectVersions",
                "rekognition:DetectCustomLabels",
                "rekognition:DetectProtectiveEquipment",
```

```
                "rekognition:ListTagsForResource",
               "rekognition:ListDatasetEntries",
                "rekognition:ListDatasetLabels",
                "rekognition:DescribeDataset"

            ],
            "Resource": "*"
        }
    ]
}
```

Ejemplo 2: Permitir al usuario acceso completo a los recursos

En el siguiente ejemplo se concede acceso completo a los recursos de Amazon Rekognition.

JSON

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

Cómo permitir a los usuarios consultar sus propios permisos

En este ejemplo, se muestra cómo podría crear una política que permita a los usuarios de IAM ver las políticas administradas e insertadas que se asocian a la identidad de sus usuarios. Esta política incluye permisos para completar esta acción en la consola o mediante programación mediante la API AWS CLI o AWS .

```
{
    "Version": "2012-10-17",
```

```
    "Statement": [
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "iam:ListAttachedUserPolicies",
                "iam:ListUserPolicies",
                "iam:GetUser"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        },
        {
            "Sid": "NavigateInConsole",
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
                "iam:ListAttachedGroupPolicies",
                "iam:ListGroupPolicies",
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
                "iam:ListUsers"
            ],
            "Resource": "*"
        }
    ]
}
```

Ejemplos de políticas de Amazon Rekognition basadas en recursos

Etiquetas personalizadas de Amazon Rekognition utiliza políticas basadas en recursos, conocidas como políticas de proyecto, destinadas a gestionar los permisos de copia de la versión de un modelo.

La política de proyecto permite o deniega el permiso para que copie la versión de un modelo de un proyecto de origen en un proyecto de destino. Necesitará una política de proyecto si el proyecto de destino está en una AWS cuenta diferente o si desea restringir el acceso dentro de una AWS cuenta. Por ejemplo, puede que desee denegar los permisos de copia a una función de IAM específica. Para obtener más información, consulte Copying a model.

Concesión de permiso para copiar una versión de modelo

En el siguiente ejemplo, se permite a la entidad principal `arn:aws:iam::123456789012:role/Admin` para que copie la versión del modelo `arn:aws:rekognition:us-east-1:123456789012:project/my_project/version/test_1/1627045542080`.

JSON

```
{
  "Version":"2012-10-17",
  "Statement":[
    {
      "Effect":"Allow",
      "Principal":{
        "AWS":"arn:aws:iam::123456789012:role/Admin"
      },
      "Action":"rekognition:CopyProjectVersion",
      "Resource":"arn:aws:rekognition:us-east-1:123456789012:project/my_project/
version/test_1/1627045542080"
    }
  ]
}
```

Solución de problemas de identidad y acceso de Amazon Rekognition

Utilice la siguiente información para diagnosticar y solucionar los problemas comunes que es posible que surjan cuando se trabaja con Amazon Rekognition e IAM.

Temas

- No tengo autorización para realizar una acción en Amazon Rekognition
- No estoy autorizado a realizar lo siguiente: PassRole
- Soy administrador y deseo permitir que otros accedan a Amazon Rekognition
- Quiero permitir que personas ajenas a mi AWS cuenta accedan a mis recursos de Amazon Rekognition

No tengo autorización para realizar una acción en Amazon Rekognition

Si Consola de administración de AWS le indica que no está autorizado a realizar una acción, debe ponerse en contacto con su administrador para obtener ayuda. El administrador es la persona que le proporcionó las credenciales de inicio de sesión.

En el siguiente ejemplo, el error se produce cuando el usuario de IAM, `mateojackson`, intenta utilizar la consola para ver detalles sobre una *`widget`*, pero no tiene permisos `rekognition:`*`GetWidget`*.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 rekognition:GetWidget on resource: my-example-widget
```

En este caso, Mateo pide a su administrador que actualice sus políticas de forma que pueda obtener acceso al recurso *`my-example-widget`* mediante la acción `rekognition:`*`GetWidget`*.

No estoy autorizado a realizar lo siguiente: PassRole

Si recibe un error que indica que no tiene autorización para realizar la acción `iam:PassRole`, sus políticas deben actualizarse para permitirle pasar un rol a Amazon Rekognition.

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 la función al servicio.

En el siguiente ejemplo, el error se produce cuando un usuario de IAM denominado «`marymajor`» intenta utilizar la consola para llevar a cabo una acción en Amazon Rekognition. 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 AWS administrador. El administrador es la persona que le proporcionó las credenciales de inicio de sesión.

Soy administrador y deseo permitir que otros accedan a Amazon Rekognition

Para permitir que otras personas accedan a Amazon Rekognition, debe conceder permiso a los usuarios o las aplicaciones que lo necesiten. Si usa AWS IAM Identity Center para administrar las personas y las aplicaciones, debe asignar conjuntos de permisos a los usuarios o grupos para definir su nivel de acceso. Los conjuntos de permisos crean políticas de IAM y las asignan a los roles de IAM asociados a la persona o aplicación de forma automática. Para obtener más información, consulte la sección Conjuntos de permisos en la Guía del usuario de AWS IAM Identity Center .

Si no utiliza IAM Identity Center, debe crear entidades de IAM (usuarios o roles) para las personas o aplicaciones que necesitan acceso. A continuación, debe asociar una política a la entidad que les conceda los permisos correctos en Amazon Rekognition. Una vez concedidos los permisos, proporcione las credenciales al usuario o al desarrollador de la aplicación. Utilizarán esas credenciales para acceder a AWS. Para obtener más información sobre la creación de usuarios, grupos, políticas y permisos de IAM, consulte Identidades de IAM y Políticas y permisos en IAM en la Guía del usuario de IAM.

Quiero permitir que personas ajenas a mi AWS cuenta accedan a mis recursos de Amazon Rekognition

Se puede crear un rol que los usuarios de otras cuentas o las personas externas a la organización puedan utilizar para acceder a sus recursos. Se puede especificar una persona de confianza para que asuma el rol. En el caso de los servicios que respaldan políticas basadas en recursos o listas de control de acceso (ACLs), puede usar esas políticas para permitir que las personas accedan a sus recursos.

Para obtener más información, consulte lo siguiente:

- Para saber si Amazon Rekognition admite estas características, consulte Cómo funciona Amazon Rekognition 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, consulte 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.

Protección de datos en Amazon Rekognition

El modelo de responsabilidad compartida de AWS se aplica a la protección de datos en Amazon Rekognition. Como se describe en este modelo, AWS es responsable de proteger la infraestructura global que ejecuta toda la 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, recomendamos proteger las credenciales de la Cuenta de AWS y configurar cuentas de usuario 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.
- Utiliza SSL/TLS para comunicarse con los recursos de AWS. Se recomienda el uso de TLS 1.2 y recomendamos TLS 1.3.
- Configure los registros de API y de actividad de los usuarios con AWS CloudTrail. Para obtener información sobre cómo utilizar registros de seguimiento de CloudTrail para capturar actividades de AWS, consulta Working with CloudTrail trails en la Guía del usuario de AWS CloudTrail.
- Utiliza las soluciones de cifrado de AWS, junto con todos los controles de seguridad predeterminados dentro de los servicios de 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 FIPS 140-3 al acceder a AWS a través de una interfaz de línea de comandos o una API, utiliza un punto de conexión de 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. Incluye las situaciones en las que se trabaja con Rekognition u otros Servicios de AWS a través de la consola, la API, la AWS CLI, o los SDK de AWS. 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 Rekognition utiliza el cifrado de datos para proteger los datos.

Cifrado en reposo

Amazon Rekognition Image

Imágenes

Las imágenes transferidas a las operaciones de la API de Amazon Rekognition pueden almacenarse y usarse para mejorar el servicio, a menos que haya optado por no participar visitando la página de políticas de exclusión de los servicios de IA y siguiendo el proceso que se explica allí. Las imágenes almacenadas se cifran en reposo (Amazon S3) mediante AWS Key Management Service (SSE-KMS).

Colecciones

Para las operaciones de comparación de rostros que almacenan información en una colección, el algoritmo de detección subyacente detecta primero las caras de la imagen de entrada, extrae un vector para cada cara y, a continuación, almacena los vectores faciales de la colección. Amazon Rekognition utiliza estos vectores faciales al realizar la comparación de rostros. Los vectores faciales se almacenan como una matriz de valores float y se cifran en reposo.

Amazon Rekognition Video

Videos

Para analizar un vídeo, Amazon Rekognition copia sus vídeos en el servicio para su procesamiento. El vídeo pueden almacenarse y usarse para mejorar el servicio, a menos que haya optado por no participar visitando la página de políticas de exclusión de los servicios de IA y siguiendo el proceso que se explica allí. Los vídeos se cifran en reposo (Amazon S3) mediante AWS Key Management Service (SSE-KMS).

Etiquetas personalizadas de Amazon Rekognition

Etiquetas personalizadas de Amazon Rekognition cifra sus datos en reposo.

Imágenes

Para entrenar el modelo, Etiquetas personalizadas de Amazon Rekognition hace una copia de las imágenes de entrenamiento y de prueba de origen. Las imágenes copiadas se cifran en reposo en Amazon Simple Storage Service (S3) mediante el cifrado del servidor con una AWS KMS key que proporcione o una clave de JMS propia de AWS. Etiquetas personalizadas de Amazon Rekognition solo admite claves de KMS simétricas. Las imágenes de origen no se ven afectadas. Para obtener más información, consulte Training an Amazon Rekognition Custom Labels Model.

Modelos

De forma predeterminada, Etiquetas personalizadas de Amazon Rekognition cifra los modelos entrenados y los archivos de manifiesto almacenados en buckets de Amazon S3 con SSE con un Clave propiedad de AWS. Para obtener más información, consulte Protección de los datos con el cifrado del servidor. Los resultados de la formación se escriben en el bucket especificado en el parámetro de entrada `OutputConfig` para CreateProjectVersion. Los resultados de la formación se cifran mediante los ajustes de cifrado configurados para el bucket (`OutputConfig`).

Bucket de consola

La consola Etiquetas personalizadas de Amazon Rekognition crea un bucket de Amazon S3 (bucket de consola) que puede utilizar para gestionar 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. Las etiquetas personalizadas de Amazon Rekognition bloquean el acceso público al bucket de la consola.

Rekognition Face Liveness

Todos los datos relacionados con la sesión almacenados en la cuenta del servicio de Rekognition Face Liveness están completamente cifrados en reposo. De forma predeterminada, las imágenes de referencia y auditoría se cifran con una clave propia de AWS de la cuenta de servicio. Sin embargo, puede optar por proporcionar sus propias claves de AWS KMS para cifrar estas imágenes.

Cifrado en tránsito

Los puntos de conexión de API de Amazon Rekognition 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 AWS Key Management Service concepts.

Cifrado de CMK para Face Liveness

La API CreateFaceLivenessSession incluye un parámetro opcional de `KmsKeyId`. Puede proporcionar el `id` de la clave de KMS que haya creado en su cuenta. Esta clave se utilizará para cifrar las imágenes de referencia y auditoría obtenidas con la API StartFaceLivenessSession, y con la API GetFaceLivenessSessionResults, las imágenes se descifrarán con esta clave antes de devolver los resultados. Si la solicitud de CreateFaceLivenessSession incluía una OutputConfig, las imágenes de referencia y auditoría se subirán en las rutas de Amazon S3 especificadas. Le recomendamos que habilite el cifrado del servidor (SSE-S3) en sus buckets de Amazon S3 para que los datos permanezcan cifrados en reposo.

Cuando proporciona su propia id de clave de AWS KMS, el servicio Rekognition Face Liveness obtiene permiso para utilizar la clave administrada por el cliente en nombre de la entidad principal que invoca las API. Las entidades principales (usuarios o roles) utilizados para invocar las API desde el backend del cliente (API `CreateFaceLivenessSession` y`GetFaceLivenessSessionResults`) deben tener acceso para realizar lo siguiente:

- kms:DescribeKey
- kms:GenerateDataKey

- kms:Decrypt

Privacidad del tráfico entre redes

El punto de conexión de Amazon Virtual Private Cloud (Amazon VPC) para Amazon Rekognition es una entidad lógica dentro de una VPC que permite la conectividad solo a Amazon Rekognition. Amazon VPC enruta las solicitudes a Amazon Rekognition 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 Rekognition, consulte. Uso de Amazon Rekognition con puntos de conexión de Amazon VPC

Uso de Amazon Rekognition con puntos de conexión de Amazon VPC

Si utiliza Amazon Virtual Private Cloud (Amazon VPC) para alojar sus recursos de AWS, puede establecer una conexión entre su VPC y Amazon Rekognition. Puede utilizar esta conexión para habilitar Amazon Rekognition y comunicarse con los recursos de la VPC sin pasar por la Internet pública.

Amazon VPC es un servicio de AWS que puede utilizar para lanzar recursos de AWS en una red virtual que usted defina. Con una VPC, puede controlar la configuración de la red, como el rango de direcciones IP, las subredes, las tablas de ruteo y las gateways de red. Gracias a los puntos de conexión de VPC, la red de AWS gestiona el direccionamiento entre la VPC y los servicios de AWS.

Para conectar su VPC a Amazon Rekognition, debe definir un punto de conexión de VPC de la interfaz para Amazon Rekognition. Un punto final de interfaz es una interfaz de red elástica con una dirección IP privada que sirve como punto de entrada para el tráfico destinado a un AWS servicio compatible. Con el punto de conexión, se ofrece conectividad escalable de confianza con Amazon Rekognition sin necesidad de utilizar una puerta de enlace de Internet, una instancia de Traducción de direcciones de red (NAT) o una conexión de VPN. Para obtener más información, consulte ¿Qué es Amazon VPC?) en la Guía del usuario de Amazon VPC.

AWS habilita los puntos de enlace de la VPC de interfaz. PrivateLink Esta tecnología de AWS permite la comunicación privada entre los servicios de AWS a través de una interfaz de red elástica con direcciones IP privadas.

> **Note**
>
> AWS admite todos los puntos de enlace del Estándar Federal de Procesamiento de Información (FIPS) de Amazon Rekognition. PrivateLink

Creación de puntos de conexión de la VPC para Amazon Rekognition

Puede crear dos tipos de puntos de conexión de Amazon VPC para utilizarlos con Amazon Rekognition.

- Un punto de conexión de VPC se utilizará con las operaciones de Amazon Rekognition. Para la mayoría de los usuarios, este es el tipo más adecuado.
- Un punto de conexión de VPC para las operaciones de Amazon Rekognition con puntos de conexión que cumplen la publicación 140-2 del Estándar federal de procesamiento de información (FIPS) del Gobierno de los Estados Unidos.

Para comenzar a utilizar Amazon Rekognition con la VPC, utilice la consola de Amazon VPC para crear un punto de conexión de VPC de tipo interfaz para Amazon Rekognition. Para obtener instrucciones, consulte el procedimiento "Para crear un punto de conexión de interfaz para un servicio de AWS utilizando la consola" en Creación de un punto de conexión de interfaz. Tenga en cuenta los siguientes pasos del procedimiento:

- Paso 3: En Categoría de servicio, elija Servicios de AWS.
- Paso 4: En Nombre del servicio, elija una de las siguientes opciones:
  - com.amazonaws.region.rekognition: crea un punto de conexión de VPC para las operaciones de Amazon Rekognition.
  - com.amazonaws.region.rekognition-fips: crea un punto de conexión de VPC para las operaciones de Amazon Rekognition con puntos de conexión que cumplen la publicación 140-2 del Estándar federal de procesamiento de información (FIPS) del Gobierno de los Estados Unidos.

Para obtener más información, consulte Introducción en la Guía del usuario de Amazon VPC.

Creación de una política de punto de conexión de VPC para Amazon Rekognition

Puede crear una política para los puntos de conexión de VPC de Amazon Rekognition y especificar lo siguiente:

- La entidad principal que puede realizar acciones.
- Las acciones que se pueden realizar.
- Los recursos en los que se pueden llevar a cabo las acciones.

Para obtener más información, consulte Controlar el acceso a servicios con puntos de conexión de VPC en la Guía del usuario de Amazon VPC.

La política del ejemplo siguiente permite a los usuarios que se conectan a Amazon Rekognition a través del punto de conexión de VPC llamar a la operación de la API `DetectFaces`. La política impide que los usuarios realicen otras operaciones de la API de Amazon Rekognition a través del punto de conexión de VPC.

Los usuarios aún pueden llamar a otras operaciones de la API de Amazon Rekognition desde fuera de la VPC. Para obtener información acerca de cómo denegar el acceso a las operaciones de la API de Amazon Rekognition que están fuera de la VPC, consulte Políticas de Amazon Rekognition basadas en identidades.

JSON

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

Modificación de la política de punto de conexión de VPC para Amazon Rekognition

1. Abra la consola de Amazon VPC en https://console.aws.amazon.com/vpc/.
2. Si todavía no ha creado el punto de conexión para Amazon Rekognition elija Crear punto de conexión. A continuación, seleccione com.amazonaws. *Region*.rekognition y selecciona Crear punto final.
3. En el panel de navegación, elija Puntos de conexión.
4. Selecciona com.amazonaws. ***Region***.rekognition endpoint y selecciona la pestaña Política en la mitad inferior de la pantalla.
5. Elija Editar política y realice los cambios en la política.

Validación de la conformidad para Amazon Rekognition

Auditores externos evalúan la seguridad y la conformidad de Amazon Rekognition en numerosos programas de conformidad de AWS. Estos incluyen SOC, PCI, FedRAMP, HIPAA y otros.

Para obtener una lista de los servicios de AWS en el ámbito de programas de conformidad específicos, consulte Servicios de AWS en el ámbito del programa de conformidad. Para obtener información general, consulte Programas de conformidad de AWS.

Puede descargar los informes de auditoría de terceros con AWS Artifact. Para obtener más información, consulte Descarga de informes en AWS Artifact.

Su responsabilidad de conformidad al utilizar Amazon Rekognition se determina en función de la sensibilidad de los datos, los objetivos de cumplimiento de su empresa y la legislación y los reglamentos correspondientes. AWS proporciona los siguientes recursos para ayudar con la conformidad:

- Guías de inicio rápido de seguridad y conformidad: estas guías de implementación tratan consideraciones sobre arquitectura y ofrecen pasos para implementar los entornos de referencia centrados en la seguridad y la conformidad en AWS.
- Documento técnico sobre arquitectura para seguridad y conformidad de HIPAA : en este documento técnico, se describe cómo las empresas pueden utilizar AWS para crear aplicaciones conformes con HIPAA.

- AWSRecursos de conformidad: este conjunto de manuales y guías podría aplicarse a su sector y ubicación.
- AWS Config: este servicio de AWS evalúa en qué medida las configuraciones de los recursos cumplen las prácticas internas, las directrices del sector y la normativa.
- AWS Security Hub CSPM: este servicio de AWS proporciona una vista integral de su estado de seguridad en AWS que lo ayuda a verificar la conformidad con los estándares y las prácticas recomendadas del sector de seguridad.

Resiliencia en Amazon Rekognition

La infraestructura global de AWS se compone de regiones y zonas de disponibilidad de AWS. AWS Las regiones proporcionan varias zonas de disponibilidad físicamente independientes y aisladas que se encuentran conectadas mediante redes con un alto nivel de rendimiento y redundancia, además de baja latencia. 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 centros de datos únicos o múltiples.

Para obtener más información sobre las regiones y zonas de disponibilidad de AWS, consulte Infraestructura global de AWS.

Además de la infraestructura global de AWS, Amazon Rekognition ofrece varias características que le brindan ayuda con sus necesidades de resiliencia y copia de seguridad de los datos.

Configuración y análisis de vulnerabilidades en Amazon Rekognition

La configuración y los controles de TI son una responsabilidad compartida entre AWS y usted, nuestro cliente. Para obtener más información, consulte el modelo de responsabilidad compartida de AWS.

Prevención de la sustitución confusa entre servicios

En AWS, la suplantación entre servicios puede producirse cuando un servicio (el servicio de llamadas) llama a otro servicio (el servicio llamado). El servicio que lleva a cabo las llamadas se

puede manipular para actuar en función de los recursos de otro cliente a pesar de que no debe tener los permisos adecuados, lo que da como resultado un problema de suplente confuso.

Para evitarlo, AWS proporciona herramientas que te ayudan a proteger los datos de todos los servicios cuyos directores de servicio tengan acceso a los recursos de tu cuenta.

Se recomienda utilizar las claves de contexto de condición global `aws:SourceArn` y `aws:SourceAccount` en las políticas de recursos para limitar los permisos que Amazon Rekognition concede a otro servicio para el recurso.

Si el valor de `aws:SourceArn` no contiene el ID de cuenta, como un ARN de bucket de Amazon S3, debe utilizar ambas claves para limitar los permisos. Si utiliza ambas claves y el valor de `aws:SourceArn` contiene el ID de la cuenta, el valor de `aws:SourceAccount` y la cuenta en el valor de `aws:SourceArn` deben utilizar el mismo ID de cuenta cuando se utiliza en la misma instrucción de política.

Utiliza `aws:SourceArn` si desea que solo se asocie un recurso al acceso entre servicios. Utiliza `aws:SourceAccount` si quiere permitir que cualquier recurso de esa cuenta se asocie al uso entre servicios.

El valor de `aws:SourceArn` debe ser el ARN del recurso utilizado por Rekognition, que se especifica con el siguiente formato: `arn:aws:rekognition:region:account:resource`.

El valor de `arn:User` ARN debe ser el ARN del usuario que llame a la operación de análisis de vídeo (el usuario que asume un rol).

La forma más eficaz de protegerse contra el problema del suplente confuso es utilizar la clave de contexto de condición global de `aws:SourceArn` con el ARN completo del recurso.

Si no conoce el ARN completo del recurso o si está especificando varios recursos, utilice la clave de `aws:SourceArn` con caracteres comodines (*) para las partes desconocidas del ARN. Por ejemplo, `arn:aws:rekognition:*:111122223333:*`.

Para protegerse contra el problema de suplente confuso, lleve a cabo los siguientes pasos:

1. En el panel de navegación de la consola de IAM, elija la opción Roles. La consola mostrará las funciones asociadas a su cuenta actual.
2. Elija el nombre del rol que desea modificar. El rol que modifique debe tener la política de AmazonRekognitionServiceRolepermisos. Seleccione la pestaña Relaciones de confianza.
3. Elija Editar la política de confianza.

4. En la página Editar política de confianza, sustituya la política JSON predeterminada por una política que utilice una o ambas claves contextuales `aws:SourceArn` y `aws:SourceAccount` de condición global. Consulte los siguientes ejemplos de políticas.
5. Elija Actualizar política.

El ejemplo siguiente contiene políticas de confianza que muestran cómo se pueden utilizar las claves de contexto de condición global de `aws:SourceArn` y `aws:SourceAccount` en Amazon Rekognition para evitar el problema del suplente confuso.

Si trabaja con vídeos almacenados y en streaming, puede utilizar una política como la siguiente en su rol de IAM:

Si trabaja exclusivamente con vídeo almacenado, puede utilizar una política como la siguiente en su rol de IAM (tenga en cuenta que no es necesario incluir el argumento `StringLike` que especifique el `streamprocessor`):

Seguridad de la infraestructura de Amazon Rekognition

Como servicio gestionado, Amazon Rekognition está AWS protegido por la seguridad de la red global. Para obtener información sobre los servicios AWS de seguridad y cómo se AWS protege la infraestructura, consulte AWS Seguridad 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 AWSarquitectura basado en el pilar de la seguridad.

Las llamadas a la API AWS publicadas se utilizan para acceder a Amazon Rekognition 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.

Supervisión de Amazon Rekognition

La monitorización es una parte importante del mantenimiento de la fiabilidad, la disponibilidad y el rendimiento de Amazon Rekognition y de las demás soluciones de AWS. AWS ofrece las siguientes herramientas de monitorización para vigilar a Rekognition, 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, puede CloudWatch hacer un seguimiento del uso de la CPU u otras métricas de sus instancias de Amazon EC2 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 instancias de Amazon EC2 y otras fuentes. CloudTrail CloudWatch Los registros pueden monitorear la información de los archivos de registro y notificarle cuando se alcancen 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 envían casi EventBridge 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 Guía del usuario de AWS CloudTrail.

Monitorización de Rekognition con Amazon CloudWatch

Con ella CloudWatch, puede obtener métricas para operaciones individuales de Rekognition o métricas globales de Rekognition para su cuenta. Puede usar métricas para realizar un seguimiento del estado de su solución basada en Rekognition y configurar alarmas que le notifiquen cuando una o más métricas superen un umbral definido. Por ejemplo, puede ver métricas del número de errores de servidor que se han producido o métricas del número de rostros que se han detectado. También puede consultar métricas del número de veces que se ha realizado correctamente una operación de Rekognition específica. Para ver las métricas, puedes usar Amazon CloudWatch AWS Command Line Interface, Amazon o la CloudWatch API.

También puede ver métricas globales durante un periodo de tiempo seleccionado mediante la consola de Rekognition. Para obtener más información, consulte Ejercicio 4: Consultar métricas totales (consola).

Uso de CloudWatch métricas para Rekognition

Para utilizar métricas, debe especificar la siguiente información:

- La dimensión de las métricas o ninguna dimensión. Una dimensión es un par de nombre-valor que le ayuda a identificar una métrica de forma inequívoca. Rekognition tiene una dimensión denominada Operation. Proporciona métricas para una operación específica. Si no especifica ninguna dimensión, el ámbito de la métrica se establece en todas las operaciones de Rekognition dentro de su cuenta.
- El nombre de la métrica, como `UserErrorCount`.

Puede obtener datos de monitorización para Rekognition mediante la, AWS CLI la o Consola de administración de AWS la API. CloudWatch También puede usar la CloudWatch API a través de uno de los kits de desarrollo de software de Amazon AWS (SDKs) o las herramientas de la CloudWatch API. La consola muestra una serie de gráficos basados en los datos sin procesar de la CloudWatch API. En función de sus necesidades, es posible que prefiera utilizar los gráficos que se muestran en la consola o que se recuperan de la API.

En la siguiente lista se indican algunos usos frecuentes de las métricas. Se trata de sugerencias que puede usar como punto de partida y no de una lista completa.

¿Cómo?	Métricas relevantes
¿Cómo realizo un seguimiento del número de rostros reconocidos?	Monitorice la estadística `Sum` de la métrica `DetectedFaceCount` .
¿Cómo puedo saber si mi aplicación ha alcanzado el número máximo de solicitudes por segundo?	Monitorice la estadística `Sum` de la métrica `ThrottledCount` .
¿Cómo puedo monitorizar los errores de solicitud?	Utilice la estadística `Sum` de la métrica `UserErrorCount` .
¿Cómo puedo encontrar el número total de solicitudes?	Utilice las estadísticas `ResponseTime` y `Data Samples` de la métrica `ResponseTime` . Esto incluye cualquier solicitud que genere un error. Si desea ver únicamente las llamadas a operaciones que se han realizado con éxito, use la métrica `SuccessfulRequestCount` .
¿Cómo puedo monitorizar la latencia de las llamadas a operaciones de `Rekognition` ?	Utilice la métrica `ResponseTime` .
¿Cómo puedo monitorizar cuántas veces `IndexFaces` añade correctamente rostros a las colecciones de Rekognition?	Monitorice la estadística `Sum` con la métrica `SuccessfulRequestCount` y la operación `IndexFaces` . Utilice la dimensión `Operation` para seleccionar la operación y la métrica.

Debe tener los CloudWatch permisos adecuados para monitorear Rekognition con. CloudWatch Para obtener más información, consulte Autenticación y control de acceso de Amazon CloudWatch.

Acceso a métricas de Rekognition

Los siguientes ejemplos muestran cómo acceder a las métricas de Rekognition mediante la consola, AWS CLI la y CloudWatch la API. CloudWatch

Para ver las métricas (consola)

1. Abra la consola en CloudWatch . https://console.aws.amazon.com/cloudwatch/
2. Elija Métricas, elija la pestaña Todas las métricas y, a continuación, elija Rekognition.
3. Elija Métricas sin dimensiones y, a continuación, elija una métrica.

   Por ejemplo, elija la métrica DetectedFace para medir la cantidad de rostros que se han detectado.
4. Elija un valor para el intervalo de fechas. El número de métricas se muestra en el gráfico.

Para ver las métricas de las llamadas a la operación **`DetectFaces`** que se han realizado correctamente durante un periodo de tiempo (CLI).

- Abra AWS CLI e introduzca el siguiente comando:

  ```
  aws cloudwatch get-metric-statistics --metric-name
  SuccessfulRequestCount --start-time 2017-1-1T19:46:20 --end-time
  2017-1-6T19:46:57 --period 3600 --namespace AWS/Rekognition --
  statistics Sum --dimensions Name=Operation,Value=DetectFaces --region
  us-west-2
  ```

  Este ejemplo muestra las llamadas a la operación `DetectFaces` que se han realizado correctamente durante un periodo de tiempo. Para obtener más información, consulte get-metric-statistics.

Para acceder a las métricas (CloudWatch API)

- Llamar a `GetMetricStatistics`. Para obtener más información, consulta la referencia de la CloudWatch API de Amazon.

Crear una alarma

Puede crear una CloudWatch alarma que envíe un mensaje de Amazon Simple Notification Service (Amazon SNS) cuando la alarma cambie de estado. Una alarma vigila una única métrica durante el período especificado y realiza una o varias acciones en función del valor de la métrica relativo a un determinado umbral durante una serie de períodos de tiempo. La acción es una notificación que se envía a un tema de Amazon SNS o a una política de Auto Scaling.

Las alarmas invocan acciones únicamente en caso de cambios de estado sostenidos. CloudWatch las alarmas no invocan acciones simplemente porque se encuentran en un estado determinado. El estado debe haber cambiado y debe mantenerse durante el número de periodos de tiempo especificado.

Para configurar una alarma (consola)

1. Inicie sesión en Consola de administración de AWS y abra la CloudWatch consola en https://console.aws.amazon.com/cloudwatch/.
2. Elija Crear alarma. Esto lanza el Asistente de creación de alarmas.
3. En la lista Métricas sin dimensiones, elija Métricas de Rekognition y, a continuación, elija una métrica.

   Por ejemplo, elija DetectedFaceCount para configurar una alarma para un número máximo de rostros detectados.
4. En el área Intervalo de tiempo, seleccione un valor de intervalo de fechas que incluya las operaciones de detección de rostros a las que ha llamado. Elija Siguiente.
5. Rellene Nombre y Descripción. Para Siempre que, elija >= e introduzca un valor máximo de su elección.
6. Si CloudWatch quieres enviarte un correo electrónico cuando se alcance el estado de alarma, en Cuando esta alarma:, selecciona State is ALARM. Para enviar alarmas a un tema de Amazon SNS existente, en Enviar notificación a:, elija un tema de SNS existente. Para configurar el nombre y las direcciones de correo electrónico de una nueva lista de suscripciones de correo electrónico, elija Crear tema para CloudWatch guardar la lista y mostrarla en el campo para que pueda usarla para configurar futuras alarmas.

   > **Note**
   >
   > Si utiliza Crear tema para crear un nuevo tema de Amazon SNS, debe verificar las direcciones de correo electrónico para que los destinatarios previstos puedan recibir las notificaciones. Amazon SNS envía solo mensajes de correo electrónico cuando la alarma entra en un estado de alarma. Si este cambio en el estado de la alarma se produce antes de que se verifiquen las direcciones de correo electrónico, los destinatarios no reciben una notificación.
7. Obtenga una vista previa de la alarma en la sección Vista previa de la alarma. Elija Crear alarma.

Para configurar una alarma (AWS CLI)

- Abre el comando siguiente AWS CLI e introdúcelo. Cambie el valor del parámetro `alarm-actions` de forma que haga referencia a un tema de Amazon SNS que haya creado anteriormente.

  ```
  aws cloudwatch put-metric-alarm --alarm-name UserErrors --
  alarm-description "Alarm when more than 10 user errors occur"
  --metric-name UserErrorCount --namespace AWS/Rekognition --
  statistic Average --period 300 --threshold 10 --comparison-
  operator GreaterThanThreshold --evaluation-periods 2 --alarm-actions
  arn:aws:sns:us-west-2:111111111111:UserError --unit Count
  ```

  Este ejemplo muestra cómo crear una alarma para cuando se producen más de 10 errores de usuario en 5 minutos. Para obtener más información, consulte put-metric-alarm.

Para configurar una alarma (CloudWatch API)

- Llamar a `PutMetricAlarm`. Para obtener más información, consulta Amazon CloudWatch API Reference.

CloudWatch métricas de Rekognition

Esta sección contiene información sobre las CloudWatch métricas de Amazon y la dimensión de operación disponible para Amazon Rekognition.

También puede ver una vista completa de métricas de Rekognition desde la consola de Rekognition. Para obtener más información, consulte Ejercicio 4: Consultar métricas totales (consola).

CloudWatch métricas de Rekognition

En la siguiente tabla se indican las métricas de Rekognition.

Métrica	Description (Descripción)
SuccessfulRequestCount	El número de solicitudes realizadas correctamente. El intervalo de códigos de respuesta para una solicitud realizada correctamente comprende de 200 a 299.

Métrica	Description (Descripción)
	Unidad: recuento Estadísticas válidas: `Sum,Average`
ThrottledCount	El número de solicitudes restringidas. Rekognition restringe una solicitud cuando recibe más solicitudes que el límite de transacciones por segundo de su cuenta. Si el límite establecido para su cuenta se supera con frecuencia, puede solicitar un aumento del límite. Para solicitar un aumento, consulte Límites de los servicios de AWS. Unidad: recuento Estadísticas válidas: `Sum,Average`
ResponseTime	El tiempo en milisegundos que tarda Rekognition en calcular la respuesta. Unidades: 1. Recuento para la estadística `Data Samples` 2. Milisegundos para la estadística `Average` Estadísticas válidas: `Data Samples,Average` **Note** La métrica `ResponseTime` no está incluida en el panel de métricas de Rekognition.
DetectedFaceCount	El número de rostros detectados con la operación `IndexFaces` o `DetectFaces` . Unidad: recuento Estadísticas válidas: `Sum,Average`

Métrica	Description (Descripción)
DetectedLabelCount	El número de etiquetas detectadas con la operación `DetectLabels` . Unidad: recuento Estadísticas válidas: `Sum,Average`
ServerErrorCount	El número de errores de servidor. El intervalo de códigos de respuesta de un error de servidor comprende de 500 a 599. Unidad: recuento Estadísticas válidas: `Sum, Average`
UserErrorCount	El número de errores de usuario (parámetros no válidos, imagen no válida, sin permiso, etc.). El intervalo de códigos de respuesta de un error de usuario comprende de 400 a 499. Unidad: recuento Estadísticas válidas: `Sum,Average`
MinInferenceUnits	El número mínimo de unidades de inferencia especificadas durante la solicitud de `StartProjectVersion` . Unidad: recuento Estadísticas válidas: `Average`
MaxInferenceUnits	El número máximo de unidades de inferencia especificadas durante la solicitud de `StartProjectVersion` . Unidad: recuento Estadísticas válidas: `Average`

Métrica	Description (Descripción)
DesiredInferenceUnits	El número de unidades de inferencia con las que Rekognition está haciendo la escala más grande o pequeña. Unidad: recuento Estadísticas válidas: `Average`
InServiceInferenceUnits	El número de unidades de inferencia que utiliza el modelo. Unidad: recuento Estadísticas válidas: `Average` Se recomienda utilizar la estadística Promedio para obtener el promedio de 1 minuto del número de instancias que se utilizan.

CloudWatch métricas para Rekognition Streaming

Rekognition también tiene un segundo espacio de nombres que se utiliza para las operaciones de streaming, «Rekognition Streaming». En la siguiente tabla se indican las métricas de Rekognition Streaming.

Métrica	Description (Descripción)
SuccessfulRequestCount	El número de solicitudes realizadas correctamente. El intervalo de códigos de respuesta para una solicitud realizada correctamente comprende de 200 a 299. Unidad: recuento Estadísticas válidas: `Sum,Average`
CallCount	El número de operaciones especificadas realizadas en su cuenta. Estadísticas válidas: `Sum,Average`
ThrottledCount	El número de solicitudes restringidas. Rekognition restringe una solicitud cuando recibe más solicitudes que el límite de transacciones por segundo

Métrica	Description (Descripción)
	de su cuenta. Si el límite establecido para su cuenta se supera con frecuencia, puede solicitar un aumento del límite. Para solicitar un aumento, consulte Límites de los servicios de AWS. Unidad: recuento Estadísticas válidas: `Sum,Average`
ServerErrorCount	El número de errores de servidor. El intervalo de códigos de respuesta de un error de servidor comprende de 500 a 599. Unidad: recuento Estadísticas válidas: `Sum, Average`
UserErrorCount	El número de errores de usuario (parámetros no válidos, imagen no válida, sin permiso, etc.). El intervalo de códigos de respuesta de un error de usuario comprende de 400 a 499. Unidad: recuento Estadísticas válidas: `Sum,Average`
ConcurrentSessions	La cantidad de sesiones simultáneas que esta cuenta utiliza actualmente. Unidad: recuento Estadísticas válidas: `Max,Average`

CloudWatch dimensión para Rekognition

Para recuperar métricas específicas de la operación, utilice el espacio de nombres `Rekognition` y proporcione una dimensión de operación.

Para obtener más información sobre las dimensiones, consulta Dimensiones en la Guía del CloudWatch usuario de Amazon.

CloudWatch dimensión para etiquetas personalizadas Rekognition

La siguiente tabla muestra las CloudWatch dimensiones disponibles para su uso con las etiquetas personalizadas Rekognition:

Dimensión	Description (Descripción)
ProjectName	El nombre del proyecto de Etiquetas personalizadas de Rekognition creado con `CreateProject` .
VersionName	El nombre de la versión del proyecto de Etiquetas personalizadas de Rekognition creada con `CreateProjectVersion` .

Para obtener más información sobre las dimensiones, consulta Dimensiones en la Guía del CloudWatch usuario de Amazon.

Registrar llamadas a la API Amazon Rekognition con AWS CloudTrail

Amazon Rekognition está AWS CloudTrail integrado con un servicio que proporciona un registro de las acciones realizadas por un usuario, un rol o un AWS servicio en Amazon Rekognition. CloudTrail captura todas las llamadas a la API de Amazon Rekognition como eventos. Las llamadas capturadas incluyen las llamadas desde la consola de Amazon Rekognition y las llamadas desde el código a las operaciones de la API de Amazon Rekognition. Si crea una ruta, puede habilitar la entrega continua de CloudTrail eventos a un bucket de Amazon S3, incluidos los eventos de Amazon Rekognition. 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, puede determinar la solicitud que se realizó a Amazon Rekognition, la dirección IP desde la que se realizó la solicitud, quién la realizó, cuándo se realizó y detalles adicionales.

Para obtener más información CloudTrail, consulte la Guía del AWS CloudTrail usuario.

Información de Amazon Rekognition en CloudTrail

CloudTrail está habilitada en su AWS cuenta al crear la cuenta. Cuando se produce una actividad en Amazon Rekognition, esa actividad se registra CloudTrail en un evento junto AWS con otros eventos de servicio en el historial de eventos. Puede ver, buscar y descargar los eventos recientes en su cuenta. AWS Para obtener más información, consulte Visualización de eventos con el historial de CloudTrail eventos.

Para obtener un registro continuo de los eventos de su AWS cuenta, incluidos los eventos de Amazon Rekognition, cree 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 Amazon Rekognition se CloudTrail registran y se documentan en la referencia de la API de Amazon Rekognition. Por ejemplo, las llamadas a los archivos de registro `CreateStreamProcessor` y las `DetectCustomLabels` acciones generan `CreateCollection` entradas en los archivos de registro. CloudTrail

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 usuario root o AWS Identity and Access Management (IAM).
- 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 .

La descripción de las entradas de archivos de registro de Amazon Rekognition

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 con acciones para la siguiente API: `StartLabelDetection` y`DetectLabels`.


```
{
    "Records": [

        {
            "eventVersion": "1.05",
            "userIdentity": {
                "type": "AssumedRole",
                "principalId": "AIDAJ45Q7YFFAREXAMPLE",
                "arn": "arn:aws:sts::111122223333:assumed-role/Admin/JorgeSouza",
                "accountId": "111122223333",
                "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
                "sessionContext": {
                    "sessionIssuer": {
                        "type": "Role",
                        "principalId": "AIDAJ45Q7YFFAREXAMPLE",
                        "arn": "arn:aws:iam::111122223333:role/Admin",
                        "accountId": "111122223333",
                        "userName": "Admin"
                    },
                    "webIdFederationData": {},
                    "attributes": {
                        "mfaAuthenticated": "false",
                        "creationDate": "2020-06-30T20:10:09Z"
                    }
                }
            },
            "eventTime": "2020-06-30T20:42:14Z",
```

```
            "eventSource": "rekognition.amazonaws.com",
            "eventName": "StartLabelDetection",
            "awsRegion": "us-east-1",
            "sourceIPAddress": "192.0.2.0",
            "userAgent": "aws-cli/3",
            "requestParameters": {
                "video": {
                    "s3Object": {
                        "bucket": "my-bucket",
                        "name": "my-video.mp4"
                    }
                }
            },
            "responseElements": {
                "jobId":
 "653de5a7ee03bd5083edde98ea8fce5794fcea66d077bdd4cfb39d71aff8fc25"
            },
            "requestID": "dfcef8fc-479c-4c25-bef0-d83a7f9a7240",
            "eventID": "b602e460-c134-4ecb-ae78-6d383720f29d",
            "readOnly": false,
            "eventType": "AwsApiCall",
            "recipientAccountId": "111122223333"
        },
        {
            "eventVersion": "1.05",
            "userIdentity": {
                "type": "AssumedRole",
                "principalId": "AIDAJ45Q7YFFAREXAMPLE",
                "arn": "arn:aws:sts::111122223333:assumed-role/Admin/JorgeSouza",
                "accountId": "111122223333",
                "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
                "sessionContext": {
                    "sessionIssuer": {
                        "type": "Role",
                        "principalId": "AIDAJ45Q7YFFAREXAMPLE",
                        "arn": "arn:aws:iam::111122223333:role/Admin",
                        "accountId": "111122223333",
                        "userName": "Admin"
                    },
                    "webIdFederationData": {},
                    "attributes": {
                        "mfaAuthenticated": "false",
                        "creationDate": "2020-06-30T21:19:18Z"
                    }
```

```
                }
            },
            "eventTime": "2020-06-30T21:21:47Z",
            "eventSource": "rekognition.amazonaws.com",
            "eventName": "DetectLabels",
            "awsRegion": "us-east-1",
            "sourceIPAddress": "192.0.2.0",
            "userAgent": "aws-cli/3",
            "requestParameters": {
                "image": {
                    "s3Object": {
                        "bucket": "my-bucket",
                        "name": "my-image.jpg"
                    }
                }
            },
            "responseElements": null,
            "requestID": "5a683fb2-aec0-4af4-a7df-219018be2155",
            "eventID": "b356b0fd-ea01-436f-a9df-e1186b275bfa",
            "readOnly": true,
            "eventType": "AwsApiCall",
            "recipientAccountId": "111122223333"
        }
    ]
}
```

Directrices y cuotas en Amazon Rekognition

En las siguientes secciones se indican las directrices y cuotas de Amazon Rekognition. Existen dos tipos de cuotas. Las cuotas establecidas, como el tamaño máximo de la imagen, no se pueden cambiar. Las cuotas predeterminadas que aparecen en la página de AWS Service Quotas se pueden cambiar siguiendo el procedimiento descrito en la sección Cuotas predeterminadas.

Temas

- Regiones admitidas
- Cuotas establecidas
- Cuotas predeterminadas

Regiones admitidas

Para ver una lista de las regiones de AWS en las que Amazon Rekognition está disponible, consulte Regiones de AWS y puntos de conexión en la Referencia general de Amazon Web Services.

Cuotas establecidas

En la siguiente lista figuran los límites en Amazon Rekognition que no se pueden cambiar. Para obtener información sobre los límites que puede cambiar, como las transacciones por segundo (TPS), consulte Cuotas predeterminadas.

Para conocer los límites de las etiquetas personalizadas de Amazon Rekognition, consulte Guidelines and Quotas in Amazon Rekognition Custom Labels.

Amazon Rekognition Image

- El tamaño máximo de una imagen almacenada como un objeto de Amazon S3 es de 15 MB.
- Las dimensiones máximas de la imagen para `DetectModerationLabels` son de 10 000 píxeles de ancho y alto.
- Las dimensiones máximas de la imagen para `DetectLabels` son de 10 000 píxeles de ancho y alto.
- Para que se detecte un rostro, este debe ocupar más de 40 x 40 píxeles en una imagen de 1920 x 1080 píxeles. En el caso de las imágenes de más de 1920 x 1080 píxeles, se necesitará un tamaño mínimo de rostro proporcionalmente mayor.

- Las dimensiones mínimas de la imagen son 80 píxeles para el alto y el ancho. Las dimensiones mínimas de la imagen para `DetectProtectiveEquipment` son 64 píxeles para el alto y el ancho.
- Las dimensiones máximas de la imagen para `DetectProtectiveEquipment` son de 4096 píxeles de ancho y alto.
- Para que `DetectProtectiveEquipment` detecte a una persona, esta no debe ocupar menos de 100 x 100 píxeles en una imagen de 800 x 1300 píxeles. En el caso de las imágenes de más de 800 x 1300 píxeles, se necesitará un tamaño mínimo de persona proporcionalmente mayor.
- El tamaño máximo de imagen como bytes sin procesar pasados como parámetro a una API es de 5 MB. El límite es de 4 MB para la API de `DetectProtectiveEquipment`.
- Amazon Rekognition admite los formatos de imagen PNG y JPEG. Es decir, las imágenes que proporcione como entrada a distintas operaciones de API, como `DetectLabels` e `IndexFaces`, deben estar en uno de los formatos admitidos.
- El número máximo de vectores de rostro que se pueden almacenar en una única colección de rostros es de 20 millones.
- El número máximo predeterminado de vectores de usuario que se pueden almacenar en una única colección de rostros es de 10 millones.
- El número máximo de vectores de rostro coincidentes devueltos por la API de búsqueda es de 4096.
- El número máximo de vectores de usuarios coincidentes devueltos por la API de búsqueda es de 4096.
- `DetectText` puede detectar hasta 100 palabras en una imagen.
- `DetectProtectiveEquipment` puede detectar el equipo de protección individual en un máximo de 15 personas.

Si necesita más información sobre las prácticas recomendadas para la comparación de imágenes y rostros, consulte Prácticas recomendadas para sensores, vídeos e imágenes de entrada.

Análisis masivo de Amazon Rekognition Image

- El Análisis masivo de Amazon Rekognition Image puede analizar lotes de hasta 10 000 imágenes.
- El Análisis masivo de Amazon Rekognition Image admite manifiestos de entrada con un tamaño máximo de hasta 50 MB.

Vídeo almacenado en Amazon Rekognition Video

- Amazon Rekognition Video puede analizar vídeos almacenados de hasta 10 GB de tamaño.
- Amazon Rekognition Video puede analizar vídeos almacenados de hasta 6 horas de duración.
- Amazon Rekognition Video admite un máximo de 20 trabajos simultáneas por cuenta.
- Los vídeos almacenados deben codificarse con el códec H.264. Los formatos de archivo admitidos son MPEG-4 y MOV.
- Cualquier API de Amazon Rekognition Video que analice datos de audio solo admite códecs de audio AAC.
- El periodo de tiempo de vida (TTL) de los tokens de paginación es de 24 horas. Los tokens de paginación se encuentran en el campo `NextToken` representado por las operaciones Get tales como `GetLabeldetection`.

Vídeo de streaming de Amazon Rekognition Video

- Una transmisión de entrada de Kinesis Video se puede asociar como máximo con un 1 procesador de streaming de Amazon Rekognition Video.
- Una transmisión de salida de Kinesis Data se puede asociar como máximo con un 1 procesador de streaming de Amazon Rekognition Video.
- La secuencia de entrada de Kinesis Video y la secuencia de salida de Kinesis Data asociadas a un procesador de streaming de Amazon Rekognition Video no pueden compartirlas varios procesadores.
- Cualquier API de Amazon Rekognition Video que analice datos de audio solo admite códecs de audio ACC.

Cuotas predeterminadas

Puede encontrar una lista de las cuotas predeterminadas en AWS Service Quotas. Estos límites son los predeterminados y se pueden cambiar. Para solicitar un aumento de su límite, deberá crear un caso. Para ver sus límites de cuota actuales (valores de cuota aplicados), consulte Cuotas de servicio de Amazon Rekognition. Para ver el historial de uso de TPS para las API de Amazon Rekognition Image, consulte la página de Cuotas de servicio de Amazon Rekognition y elija una operación de API específica para ver el historial de esa operación.

Temas

- Calcular el cambio de cuota de TPS
- Prácticas recomendadas para las cuotas de TPS
- Crear un caso para cambiar las cuotas de TPS

Calcular el cambio de cuota de TPS

¿Cuál es el nuevo límite que solicita? Las transacciones por segundo (TPS) son más relevantes en los momentos de máxima carga de trabajo prevista. Es importante comprender el número máximo de llamadas simultáneas a la API en momentos de máxima carga de trabajo y el tiempo de respuesta (de 5 a 15 segundos). Tenga en cuenta que 5 segundos debe ser el mínimo. A continuación se muestran dos ejemplos:

- Ejemplo 1: El número máximo de usuarios simultáneos de Face Authentication (API CompareFaces) que espero al principio de mi hora de mayor actividad es de 1000. Estas respuestas se distribuirán en un período de 10 segundos. Por lo tanto, el TPS requerido es 100 (1000/10) para la API CompareFaces en mi región correspondiente.
- Ejemplo 2: El número máximo de llamadas simultáneas de detección de objetos (API DetectLabels) que se esperan al principio de mi hora de mayor actividad es de 250. Estas respuestas se distribuirán en un período de 5 segundos. Por lo tanto, el TPS requerido es 50 (250/5) para la API DetectLabels en mi región correspondiente.

Prácticas recomendadas para las cuotas de TPS

Las prácticas recomendadas para las transacciones por segundo (TPS) incluyen suavizar los picos de tráfico, configurar los reintentos y configurar las fluctuaciones y el retroceso exponencial.

1. Suavice los picos de tráfico. Los picos de tráfico afectan al rendimiento. Para obtener el máximo rendimiento de las transacciones por segundo asignadas (TPS), utilice una arquitectura de colas sin servidor u otro mecanismo para “suavizar” el tráfico de forma que sea más coherente. Para ver ejemplos de código y referencias sobre el procesamiento de imágenes y vídeos a gran escala sin servidor con Rekognition, consulte Large scale image and video processing with Amazon Rekognition.
2. Configure los reintentos. Siga las instrucciones de the section called “Gestión de errores” para configurar los reintentos para los errores que los permiten.

3. Configure la fluctuación y el retroceso exponencial. La configuración de fluctuación y retroceso exponencial a medida que configura los reintentos le permite mejorar el rendimiento alcanzable. Consule Error retries and exponential backoff in AWS.

Crear un caso para cambiar las cuotas de TPS

Para crear un caso, vaya a Crear caso y responda a las siguientes preguntas:

- ¿Ha implementado las the section called “Prácticas recomendadas para las cuotas de TPS” para suavizar los picos de tráfico y configurar los reintentos, la fluctuación y el retroceso exponencial?
- ¿Ha calculado el cambio de cuota de TPS que necesita? Si no es así, consulte the section called “Calcular el cambio de cuota de TPS”.
- ¿Ha comprobado su historial de uso del TPS para predecir con mayor precisión sus necesidades futuras? Para ver tu historial de uso del TPS, consulte la página de cuotas de servicio de Amazon Rekognition.
- ¿Cuál es su caso de uso?
- ¿Qué API piensa usar?
- ¿En qué regiones piensa usar estas API?
- ¿Puede distribuir la carga entre varias regiones?
- ¿Cuántas imágenes procesa a diario?
- ¿Cuánto tiempo espera mantener este volumen (determina se trata de un pico puntual o es continuo)?
- ¿Cómo te bloquea el límite predeterminado? Revise la siguiente tabla de excepciones para confirmar la situación en la que se encuentra.

Código de error	Excepción	Mensaje	¿Qué significa?	¿Se puede reintentar?
Código de estado HTTP 400	ProvisionedThroughputExceededException	Se ha superado la tasa aprovisionada.	Indica limitaciones. Puede volver a intentar o evaluar una solicitud de	Sí

Código de error	Excepción	Mensaje	¿Qué significa?	¿Se puede reintentar?
			aumento de límite.	
Código de estado HTTP 400	Throttlin gException	Vaya más despacio; aumento repentino de la tasa de solicitud es.	Es posible que esté enviando un tráfico intenso y se encuentre con una limitació n. Debería dar forma al tráfico y hacerlo más fluido y consistente. A continuac ión, configure los reintento s adicional es. Consulte las prácticas recomendadas.	Sí
Código de estado HTTP 5xx	Throttlin gException (HTTP 500)	Servicio no disponible	Indica que el backend se está ampliando para soportar la acción. Debería volver a intentar la solicitud.	Sí

Para obtener información detallada sobre los códigos de error, consulte the section called “Gestión de errores”.

> **Note**
>
> Estos límites dependen de la región en la que se encuentre. Si opta por cambiar un límite, afectará a la operación de API que solicite, en la región en la que la solicite. El resto de las operaciones y las regiones de la API no se ven afectadas.

Historial de documentos de Amazon Rekognition

En la siguiente tabla se describen los cambios importantes en cada versión de la Guía para desarrolladores de Amazon Rekognition. Para recibir notificaciones sobre los cambios en esta documentación, puede suscribirse a una fuente RSS.

- Última actualización de la documentación: 15 de junio de 2023

Cambio	Descripción	Fecha
Amazon Rekognition admite ahora nuevas etiquetas de moderación y ofrece precisión mejorada para la moderación de contenido de imágenes.	Se ha mejorado la función de moderación de contenido de Amazon Rekognition para mejorar la precisión, la detección de nuevas etiquetas y la capacidad de identificar contenido ilustrado animado and/or .	1 de febrero de 2024
Amazon Rekognition ahora admite el análisis masivo de imágenes	Amazon Rekognition ahora admite el procesamiento de una gran colección de imágenes de forma asíncrona mediante un archivo de manifiesto con la operación. StartMediaAnalysisJob	23 de octubre de 2023
Amazon Rekognition ahora admite la moderación de contenido personalizada con adaptadores	Amazon Rekognition ahora admite una mayor precisión de la API mediante el uso de adaptadores que amplían las capacidades DetectModerationLabels de los modelos de aprendizaje profundo de Rekognition existentes.	12 de octubre de 2023

Rekognition ahora admite vectores de usuario con colecciones	Las colecciones de rostros de Rekognition ahora admiten la creación de vectores de usuario. Los vectores de usuario agregan varios vectores faciales del mismo usuario, lo que mejora la precisión con represent aciones más sólidas de un usuario.	12 de junio de 2023
Se han agregado acciones para implicar la administr ación de los usuarios a las siguientes políticas administr adas: AmazonRekognitionR eadOnlyAccess	Amazon Rekognition agregó las siguientes acciones a las políticas administr adas de `AmazonRek ognitionReadOnlyAc cess` : `ListUsers` , `SearchUsers` y `SearchUse rsByImage`	12 de junio de 2023
Amazon Rekognition Image ahora puede deducir la dirección de la mirada	Se han realizado mejoras en las operaciones de detección de rostros de Amazon Rekognition Image, que ahora pueden deducir la dirección de la mirada en un rostro detectado.	31 de mayo de 2023

Se mejoró la API de moderación de contenido de Rekognition	Rekognition mejoró el modelo de moderación de contenido para la moderación de imágenes y vídeos. La mejora amplía considerablemente la detección de contenido explícito, violento y sugerente . Los clientes ahora pueden detectar el contenido explícito y violento con mayor precisión para mejorar la experienc ia del usuario final, proteger la identidad de su marca y garantizar que todo el contenido cumpla con las normas y políticas del sector.	9 de mayo de 2023
Amazon Rekognition Image ahora puede detectar rostros ocultos	Amazon Rekognition Image ahora puede detectar la oclusión de rostros. Amazon Rekognition DetectFaces Image devuelve un nuevo FaceOccluded atributo, que indica si el rostro IndexFace s APIs de una imagen está parcialmente capturado o no es totalmente visible debido a la superposición de objetos, ropa y partes del cuerpo.	5 de mayo de 2023

Rekognition ahora puede detectar pruebas de vida del rostro	Amazon Rekognition Video ahora se puede utilizar para detectar la intensidad de un vídeo y comprobar que el usuario que está delante de una cámara está físicamen te presente. El detector Face Liveness también detecta los ataques simulados que se presentan ante una cámara o cuando se intenta esquivar una cámara.	11 de abril de 2023
Actualización a Amazon Rekognition Video.	Amazon Rekognition Video ahora puede detectar más etiquetas y devolver más información sobre los atributos de las imágenes y las etiquetas. La GetLabelD etection API ahora devuelve información sobre los alias y las categorías. La informaci ón de la etiqueta devuelta se puede filtrar con opciones de filtro inclusivas y exclusiva s. Los resultados se pueden agregar por marcas de tiempo o segmentos de vídeo.	7 de diciembre de 2022

Actualización a Amazon Rekognition Image.	Amazon Rekognition Image ahora puede detectar más etiquetas y devuelve más información sobre los atributos de las imágenes y las etiquetas. La DetectLabels API ahora devuelve información sobre los alias, las categoría s y las propiedades de las imágenes, como los colores dominantes. La información de la etiqueta devuelta se puede filtrar con opciones de filtro inclusivas y exclusivas.	11 de noviembre de 2022
Se han agregado acciones para Model ProjectPolicy Copy y etiquetas personalizadas a las siguientes políticas gestionadas: AmazonRek ognitionReadOnlyAccess	Amazon Rekognition ha añadido las siguientes acciones a la política administr ada `AmazonRekognitionR eadOnlyAccess` : `ListProjectPolicies`	21 de julio de 2022
Se han agregado acciones ProjectPolicy y Custom Labels Model Copy a las siguientes políticas gestionad as: AmazonRekognitionF ullAccess, AmazonRek ognitionCustomLabelsFullAcc ess	Rekognition agregó las siguientes acciones a las políticas administradas `AmazonRekognitionC ustomLabelsFullAcc ess` y `AmazonRek ognitionFullAccess` : `CopyProjectVersion` , `PutProjectPolicy` , `ListProjectPolicies` y `DeleteProjectPolicy`	21 de julio de 2022

Amazon Rekognition Video ahora puede detectar etiquetas en la transmisión de vídeo	Amazon Rekognition Video puede detectar etiquetas como mascotas y paquetes en la transmisión de vídeo. Esto se hace mediante la opción de configuración `ConnectedHome` de los procesadores de transmisión creados con la operación `CreateStreamProcessor` .	28 de abril de 2022
La referencia a la API se ha retirado de la guía para desarrolladores de Amazon Rekognition	La referencia de la API de Amazon Rekognition ya está disponible en la Referencia de API de Amazon Rekognition.	24 de febrero de 2022
Actualización de la administración del conjunto de datos para las siguientes políticas administradas:Política gestionada por AWS: AmazonRekognitionReadOnlyAccess,Política gestionada por AWS: AmazonRekognitionFullAccess, Política gestionada por AWS: AmazonRekognitionCustomLabelsFullAccess	Amazon Rekognition agregó las siguientes acciones `AmazonRekognitionReadOnlyAccess` a `AmazonRekognitionFullOnlyAccess` las `CreateDataset` políticas y administróListDatasetEntries : `AmazonRekognitionCustomLabelsFullAccess` `ListDatasetLabels` ,,,,,, `DescribeDataset` `UpdateDatasetEntries` `DistributeDatasetEntries` DeleteDataset	1 de noviembre de 2021

Un nuevo nodo de la tabla de contenido muestra ejemplos de Amazon Rekognition alojados en GitHub	Los ejemplos de código actualizados del repositorio de ejemplos de AWS código ahora aparecen en un nodo independiente de la guía para desarrolladores de Amazon Rekognition para facilitar el acceso.	22 de octubre de 2021
Amazon Rekognition puede detectar fotogramas negros y contenido principal del programa en segmentos de vídeo	Amazon Rekognition puede identificar los fotograma s negros, las barras de color, los créditos iniciales , los créditos finales, los logotipos de los estudios y el contenido principal del programa como indicadores técnicos de un vídeo mediante las operaciones `StartSegm entDetection` y `GetSegmentDetection` .	7 de junio de 2021
Actualización de la administr ación de conjuntos de datos para las siguientes políticas administradas:	Puede utilizar la operación `DetectText` de Amazon Rekognition para detectar hasta 100 palabras en una imagen.	21 de mayo de 2021
Actualización de etiquetado para y AmazonRekognitionR eadOnlyAccessAmazonRek ognitionFullAccess	Rekognition agregó nuevas acciones de etiquetado a las políticas `AmazonRek ognitionFullAccess` y `AmazonRekognitionR eadOnlyAccess` .	2 de abril de 2021

Amazon Rekognition ahora admite el etiquetado	Ahora puede usar etiquetas para identificar, organizar, buscar y filtrar las colecciones, los procesadores de streaming y los modelos de etiquetas personalizadas de Amazon Rekognition.	25 de marzo de 2021
Amazon Rekognition ahora puede detectar equipos de protección individual	Amazon Rekognition ahora puede detectar si las personas que aparecen en una imagen se cubren las manos, la cara y la cabeza.	15 de octubre de 2020
Amazon Rekognition tiene nuevas categorías de moderación de contenido	Las categorías de moderació n de contenido de Amazon Rekognition ahora incluyen 6 nuevas categorías: drogas, tabaco, alcohol, juegos de azar, gestos groseros y símbolos de odio.	12 de octubre de 2020
Nuevo tutorial para mostrar localmente los resultados de Amazon Rekognition Video Streams de Kinesis Video Streams	Puede mostrar la salida de Amazon Rekognition Video de un vídeo en streaming en Kinesis Video Streams en una transmisión de vídeo local.	20 de julio de 2020
Nuevo tutorial de Amazon Rekognition para usar Gstreamer	Con Gstreamer, puede incorporar una transmisión de vídeo en directo desde la cámara de un dispositivo a Amazon Rekognition Video a través de Kinesis Video Streams.	17 de julio de 2020

Amazon Rekognition ahora admite la segmentación de vídeos almacenados	Con la API de segmentación asíncrona de Amazon Rekognition Video puede detectar fotogramas negros, barras de color, créditos finales y tomas en vídeos almacenados.	22 de junio de 2020
Amazon Rekognition ahora es compatible con las políticas de punto de conexión de VPC de Amazon	Al especificar una política, puede restringir el acceso a un punto de conexión de VPC de Amazon de Amazon Rekognition.	3 de marzo de 2020
Amazon Rekognition ahora admite la detección de texto en los vídeos almacenados	Puede utilizar la API de Amazon Rekognition Video para detectar de forma asíncrona el texto de un vídeo almacenado.	17 de febrero de 2020
Amazon Rekognition ya es compatible con la inteligencia artificial aumentada (vista previa) y las etiquetas personalizadas de Amazon Rekognition	Con Amazon Rekognition Custom Labels, puede detectar objetos, escenas y conceptos especializados en las imágenes mediante la creación de su propio modelo de aprendizaje automático. DetectModerationLabels ahora es compatible con Amazon Augmented AI (versión preliminar).	3 de diciembre de 2019
Amazon Rekognition ahora es compatible con AWS PrivateLink	Con AWS PrivateLink , puede establecer una conexión privada entre su VPC y Amazon Rekognition.	12 de septiembre de 2019

Filtrado facial de Amazon Rekognition	Amazon Rekognition añade una compatibilidad mejorada con el filtrado de rostros IndexFaces a la operación de la API e introduce el filtrado CompareFaces de rostros para las operaciones de la API. SearchFacesByImage	12 de septiembre de 2019
Ejemplos de Amazon Rekognition Video actualizados	Código de ejemplo de Amazon Rekognition Video actualizado para crear y configurar el tema de Amazon SNS y la cola de Amazon SQS.	5 de septiembre de 2019
Se han añadido ejemplos de Ruby y Node.js	Se han añadido ejemplos de Amazon Rekognition Image de Ruby y Node.js para la detección sincrónica de rostros y etiquetas.	19 de agosto de 2019
Se ha actualizado la detección de contenido no seguro	La detección de contenido no seguro de Amazon Rekogniti on ahora puede detectar contenido violento.	9 de agosto de 2019
GetContentModeration operación actualizada	GetContentModeration ahora devuelve la versión del modelo de detección de moderació n utilizado para detectar contenido no seguro.	13 de febrero de 2019

GetLabelDetection y DetectModerationLabels operaciones actualizadas	GetLabelDetection ahora devuelve información sobre los cuadros delimitadores de los objetos comunes y una taxonomía jerárquica de las etiquetas detectadas. Ahora se devuelve la versión del modelo utilizada para la detección de etiquetas . DetectModerationLabels ahora devuelve la versión del modelo utilizada para detectar contenido no seguro.	17 de enero de 2019
DetectFaces y IndexFaces operación actualizada	Esta versión actualiza la IndexFaces operación DetectFaces y. Si el parámetro de entrada Attributes está establecido en TODOS, los puntos de referencia de ubicación de la cara incluyen 5 puntos de referencia nuevos: upperJawlineLeft, midJawlin eLeft, ChinBottom,, midJawlin eRight. upperJawlineRight	19 de noviembre de 2018
DetectLabels operación actualizada	Ahora, se devuelven cuadros delimitadores para algunos objetos. Está disponible la taxonomía jerárquica para las etiquetas. Ya se puede obtener la versión del modelo de detección utilizado en la detección.	1 de noviembre de 2018

IndexFaces operación actualizada	Ahora IndexFaces puede utilizar el parámetro QualityFilter de entrada para filtrar los rostros detectados con baja calidad. También puede utilizar el parámetro MaxFaces de entrada para reducir el número de rostros devueltos en función de la calidad de la detección de rostros y del tamaño del rostro detectado.	18 de septiembre de 2018
DescribeCollection operación agregada	Ahora puede obtener información sobre una colección existente llamando a la DescribeCollection operación.	22 de agosto de 2018
Nuevos ejemplos de Python	Se han añadido ejemplos de Python al contenido de Amazon Rekognition Video y se ha reorganizado parte del contenido.	26 de junio de 2018
Diseño del contenido actualizado	El contenido de Amazon Rekognition Image se ha reorganizado y se han añadido nuevos ejemplos de Python y de C#.	29 de mayo de 2018

Amazon Rekognition admite AWS CloudTrail	Amazon Rekognition está AWS CloudTrail integrado con un servicio que proporciona un registro de las acciones realizadas por un usuario, un rol o un AWS servicio en Amazon Rekognition. Para obtener más información, consulte Registro de llamadas a la API Amazon Rekognition con AWS. CloudTrail	6 de abril de 2018
Analizar vídeos almacenados y en streaming. Nuevo índice	Para obtener informaci ón sobre el análisis de los vídeos almacenados, consulte Trabajar con vídeos almacenados. Para obtener información sobre el análisis de los vídeos en streaming, consulte Trabajar con vídeos en streaming. El índice para la documentación de Amazon Rekognition se ha reordenado para dar cabida a las operaciones de imagen y vídeo.	29 de noviembre de 2017

Modelos de detección de texto en imágenes y rostros	Amazon Rekognition ahora puede detectar texto en imágenes. Para obtener más información, consulte Detecting Text. Amazon Rekognition introduce el control de versiones para el modelo de aprendizaje profundo de detección de rostros. Para obtener más información, consulte Control de versiones del modelo.	21 de noviembre de 2017
Reconocimiento de famosos	Ahora Amazon Rekognition puede analizar imágenes para detectar famosos. Para obtener más información, consulte Reconocimiento de famosos.	8 de junio de 2017
Moderación de imágenes	Amazon Rekognition ahora puede determinar si una imagen incluye contenido para adultos explícito o insinuante. Para obtener más información, consulte Detección de contenido no seguro.	19 de abril de 2017

Rango de edades para los rostros detectados. Panel de métricas globales de Rekognition	Amazon Rekognition ahora devuelve una estimación del rango de edades, en años, para los rostros detectados por la API de Amazon Rekognition. Para obtener más información, consulte AgeRange. La consola de Rekognition ahora tiene un panel de métricas que muestra los gráficos de actividad de un conjunto de métricas de Amazon CloudWatch para Rekognition durante un período de tiempo específico. Para obtener más información, consulte Ejercicio 4: Consultar métricas totales (consola).	9 de febrero de 2017
Nuevo servicio y guía	Esta es la versión inicial del servicio de análisis de imágenes, Amazon Rekognition y la Guía para desarrolladores de Amazon Rekognition.	30 de noviembre de 2016

AWSGlosario de

Para ver la terminología más reciente de AWS, consulte el Glosario de AWS en la Referencia de Glosario de AWS.

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.