
Amazon Rekognition

Guía para desarrolladores



Amazon Rekognition: Guía para desarrolladores

Copyright © 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 menoscalice o desacredite a Amazon. Todas las demás marcas comerciales que no sean 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
Elegibilidad de Amazon Rekognition y HIPAA	3
¿Es la primera vez que usa Amazon Rekognition?	3
Cómo funciona	4
Tipos de análisis	4
Etiquetas	4
Etiquetas personalizadas	5
Rostros	5
Búsqueda de rostros	5
Recorridos del personal	5
Equipo de protección personal	6
Famosos	6
Detección de texto	6
Contenido inapropiado u ofensivo	6
Operaciones de imágenes y vídeo	6
Operaciones de Amazon Rekognition Image	6
Operaciones de Amazon Rekognition Video	7
Operaciones sin almacenamiento de información y basadas en almacenamiento de información	7
Uso del SDK de AWS o HTTP para llamar a las operaciones de API de Amazon Rekognition	7
Operaciones de API sin almacenamiento y almacenamiento	8
Operaciones sin almacenamiento	8
Operaciones de API basadas en almacenamiento de información	9
Control de versiones del modelo	10
Introducción	12
Paso 1: Configurar una cuenta de	12
Inscripción en AWS	12
Creación de un usuario de IAM	13
Paso siguiente	13
Paso 2: Configurar la AWS CLI y AWSSDK de	14
Paso siguiente	15
Uso de Rekognition con un AWSSDK	15
Paso 3: Primeros pasos con AWS CLI y AWSSDK API	15
Formato del AWS CLI Ejemplos de	16
Paso siguiente	16
Paso 4: Primeros pasos con la consola	16
Ejercicio 1: Detección de objetos y escenas (consola)	17
Ejercicio 2: Analizar caras (consola)	22
Ejercicio 3: Comparar caras (consola)	24
Ejercicio 4: Ver métricas globales (consola)	26
Trabajo con imágenes y vídeos	28
Trabajar con imágenes	28
Especificaciones de imágenes	29
Análisis de imágenes en un bucket de Amazon S3	30
Uso de un sistema de archivos local	39
Visualización de cuadros delimitadores	49
Obtener orientación de imagen y coordenadas de cuadro delimitador	57
Trabajar con vídeos almacenados	63
Tipos de análisis	64
Descripción general de la API de Amazon Rekognition Video	64
Llamar a las operaciones de Amazon Rekognition Video	66
Configuración de Amazon Rekognition Video	70
Análisis de un vídeo almacenado (SDK)	72
Análisis de un vídeo (AWS CLI)	89
Referencia: Notificación de los resultados del análisis	91

Solución de problemas de Amazon Rekognition Video	92
Uso de vídeos en streaming	94
Configuración de los recursos de Amazon Rekognition Video y Amazon Kinesis	95
Transmisión mediante un complemento de GStreamer	107
Lectura de resultados del análisis	109
Referencia: Registro de reconocimiento facial de Kinesis	113
Solución de problemas de streaming de vídeo	117
Control de errores	122
Componentes del error	122
Mensajes y códigos de error	123
Administración de errores en la aplicación	127
Uso de Amazon Rekognition con FedRAMP	127
Prácticas recomendadas para sensores, imágenes de entrada y vídeos	131
Latencia de operación de imagen de Amazon Rekognition	131
Recomendaciones para imágenes de entrada de comparación facial	131
Recomendaciones para la configuración de la cámara (imagen y vídeo)	132
Recomendaciones para la configuración de la cámara (almacenado y streaming de vídeo)	133
Recomendaciones para la configuración de la cámara (streaming de vídeo)	134
Detección de etiquetas	135
Cuadros delimitadores y etiquetas padre	135
Detección de etiquetas en una imagen	136
Solicitud de operación DetectLabels	142
Respuesta DetectLabels	143
Detección de etiquetas en un vídeo	146
Respuesta de la operación GetLabelDetection	147
Detección de etiquetas personalizadas	148
Detección y análisis de rostros	149
Información general de la detección y comparación de rostros	150
Directrices sobre los atributos faciales	150
Detección de rostros en una imagen	151
Solicitud de operación DetectFaces	158
Respuesta de la operación DetectFaces	158
Comparación de rostros en imágenes	163
Solicitud de operación CompareFaces	170
Respuesta de la operación CompareFaces	170
Detección de rostros en un vídeo almacenado	172
Respuesta de la operación GetFaceDetection	177
Búsqueda de rostros en una colección	181
Administración de colecciones	181
Administración de rostros en una colección	182
Directrices para usar IndexFaces	182
Aplicaciones críticas o de seguridad pública	182
Aplicaciones para compartir fotos y redes sociales	182
Uso general	182
Búsqueda de caras dentro de una colección	182
Uso de umbrales de similitud para hacer coincidir caras	183
Casos de uso que implican seguridad pública	183
Uso de Amazon Rekognition para ayudar a la seguridad pública	185
Creación de una colección	185
Solicitud de operación CreateCollection	189
Respuesta de la operación CreateCollection	189
Etiquetado de colecciones	189
Incorporación de etiquetas en una colección nueva	189
Incorporación de etiquetas en una colección existente	190
Listar etiquetas de una colección	191
Eliminar etiquetas de una colección	191
Listado de colecciones	192

Solicitud de operación ListCollections	196
Respuesta de operación ListCollections	196
Descripción de una colección	196
Solicitud de operación DescribeCollection	200
Respuesta de operación DescribeCollection	200
Eliminación de una colección	201
Solicitud de operación DeleteCollection	204
Respuesta de operación DeleteCollection	204
Incorporación de rostros en una colección	204
Filtrado de rostros	205
Solicitud de operación IndexFaces	210
Respuesta de la operación IndexFaces	211
Listado de rostros en una colección	216
Solicitud de operación ListFaces	220
Respuesta de la operación ListFaces	220
Eliminación de rostros de una colección	221
Solicitud de operación DeleteFaces	224
Respuesta de la operación DeleteFaces	224
Búsqueda de un rostro (ID de cara)	224
Solicitud de operación SearchFaces	228
Respuesta de la SearchFaces	228
Búsqueda de un rostro (imagen)	229
Solicitud de operación SearchFacesByImage	233
Respuesta de la operación SearchFacesByImage	233
Búsqueda de rostros en vídeos almacenados	234
Respuesta de la operación GetFaceSearch	239
Recorridos de las personas	243
Respuesta de la operación GetPersonTracking	247
Detección de equipos de protección personal	251
Tipos de EPP	252
Cubierta de rostros	252
Funda de mano	252
Tapa de cabeza	252
Fianza de detección de EPP	252
Resumen de EPP detectado en una imagen	252
Tutorial: Creación de un valorAWS Lambdafunción que detecta imágenes con EPP	253
Descripción de la API de detección de EPP	253
Suministro de una imagen	253
Descripción de la respuesta de DetectProtective Equipment	254
Detección de EPP en una imagen	258
Ejemplo: cuadros delimitadores y cubiertas de caras	265
Reconocimiento de famosos	275
Reconocimiento de celebridades en comparación con la búsqueda facial	275
Reconocimiento de famosos en una imagen	276
Llamar a RecognizeCelebrities	276
Solicitud de operación RecognizeCelebrities	282
Respuesta de la operación RecognizeCelebrities	282
Reconocimiento de famosos en un vídeo almacenado	284
Respuesta de operación GetCelebrityRecognition	292
Obtención de información de famosos	294
Llamar a GetCelebrityInfo	294
Solicitud de operación GetCelebrityInfo	297
Respuesta de la operación GetCelebrityInfo	297
Moderación de contenido	298
Uso de las API de moderación de imagen y vídeo	298
Detección de imágenes inapropiadas	300
Detección de contenido inadecuado en una imagen	300

.....	300
Solicitud de operación DetectModerationLabels	304
Respuesta de la operación DetectModerationLabels	304
Detección de vídeos almacenados inapropiados	305
Respuesta de la operación GetContentModeration	310
Revisión de contenido inadecuado con Amazon A2I	311
Detección de texto	315
Detección de texto en una imagen	316
Solicitud de operación DetectText	321
Respuesta de la operación DetectText	321
Detección de texto en un vídeo almacenado	328
Filtros	333
Respuesta GetTextDetection	333
Detección de segmentos de vídeo	338
Tomas técnicas	339
Fotogramas negros	339
Créditos	339
Barras de color	339
Secuencias	339
Logos de Studio	339
Contenidos	340
Detección de tomas	340
Acerca de la API de detección de segmentos de Amazon Rekognition Video	341
Uso de la API de segmentos de Amazon Rekognition	341
Inicio del análisis de segmentos	341
Obtener resultados de análisis de segmentos	342
Ejemplo: Detección de segmentos en un vídeo almacenado	346
Tutoriales	354
Almacenamiento de datos de Amazon Rekognition con Amazon RDS y DynamoDB	354
Requisitos previos	354
Obtención de etiquetas para imágenes en un depósito de Amazon S3	355
Creación de una tabla de Amazon DynamoDB	356
Carga de datos en DynamoDB	357
Creación de una base de datos MySQL en Amazon RDS	358
Carga de datos en una tabla MySQL de Amazon RDS	359
Uso de Amazon Rekognition y Lambda para etiquetar activos en un bucket de Amazon S3	361
Requisitos previos	361
Configurar el rol Lambda de IAM	362
Creación del proyecto	362
Escribir el código	364
Empaquetar el proyecto	371
Implemente la función Lambda	371
Probar el método Lambda	371
CrearAWS aplicaciones de analizador de vídeo	372
Requisitos previos	373
Procedimiento	373
Creación de una función de Lambda de Amazon Rekognition	373
Requisitos previos	374
Creación del tema de SNS	374
Crear la función de Lambda	375
Configure la función Lambda	375
Configurar el rol Lambda de IAM	376
CrearlaAWS Toolkit for Eclipse Proyecto Lambda	376
Probar la función de Lambda	379
Uso de Amazon Rekognition para la verificación de identidad	380
Requisitos previos	380
Creación de una colección	381

Registro de nuevos usuarios	382
Inicio de sesión del usuario existente	387
Ejemplos de código	390
Ejemplos de servicios combinados	391
Detección de EPP en imágenes	391
Detectar rostros en una imagen	392
Detección de objetos en imágenes	393
Detección personas y objetos en un vídeo	394
Guardar EXIF y otra información de imagen	395
Ejemplos de uso	396
Construye una colección y encuentra caras en ella	396
Detección y visualización de elementos en imágenes	403
Detectar información en vídeos	412
Ejemplos de API	430
Comparar rostros de una imagen con una imagen de referencia	430
Crear una colección	435
Eliminar una colección	437
Eliminación de rostros de una colección	439
Describir una colección	442
Detectar rostros en una imagen	445
Detectar etiquetas en una imagen	450
Detectar etiquetas de moderación en una imagen	454
Detectar texto en una imagen	457
Obtener información sobre famosos	460
Indexar rostros en una colección	461
Listado de colecciones	465
Mostrar una lista de rostros en una colección	468
Reconocer famosos en una imagen	471
Búsqueda de rostros en una colección	474
Búsqueda de caras de una colección en comparación con una imagen de referencia	477
Seguridad	481
Identity and Access Management	481
Público	481
Autenticación con identidades	482
Administración de acceso mediante políticas	484
Cómo funciona Amazon Rekognition con IAM	486
Políticas administradas por AWS	489
Uso de ejemplos de políticas basadas en identidades	493
Solución de problemas	496
Protección de los datos	498
Cifrado de datos	499
Privacidad del tráfico entre redes	500
Monitoreo	500
Uso de métricas de CloudWatch para Rekognition	500
Acceso a las métricas de reconocimiento	501
Crear una alarma	502
Métricas de CloudWatch para Rekognition	503
Registro de llamadas a la API de Amazon Rekognition con AWS CloudTrail	504
Información de Amazon Rekognition en CloudTrail	505
Descripción de las entradas de archivos de registro de Amazon Rekognition	505
Uso de Amazon Rekognition con endpoints de Amazon VPC	507
Creación de puntos de enlace de Amazon VPC para Amazon Rekognition	508
Creación de una política de punto de enlace de la VPC para Amazon Rekognition	508
Validación de conformidad	509
Resiliencia	510
Configuración y análisis de vulnerabilidades	510
Seguridad de infraestructuras	510

Referencia de la API	511
Amazon Rekognition Image	511
Etiquetas personalizadas de Amazon Rekognition	511
Vídeo almacenado de Amazon Rekognition	512
Amazon Rekognition Video Streaming de vídeo	512
Encabezados HTTP	512
Encabezados HTTP	512
Acciones	513
CompareFaces	515
CreateCollection	522
CreateDataset	525
CreateProject	529
CreateProjectVersion	532
CreateStreamProcessor	537
DeleteCollection	541
DeleteDataset	543
DeleteFaces	545
DeleteProject	548
DeleteProjectVersion	551
DeleteStreamProcessor	554
DescribeCollection	556
DescribeDataset	559
DescribeProjects	561
DescribeProjectVersions	564
DescribeStreamProcessor	569
DetectCustomLabels	573
DetectFaces	578
DetectLabels	583
DetectModerationLabels	588
DetectProtectiveEquipment	592
DetectText	596
DistributeDatasetEntries	600
GetCelebrityInfo	602
GetCelebrityRecognition	605
GetContentModeration	611
GetFaceDetection	615
GetFaceSearch	620
GetLabelDetection	626
GetPersonTracking	630
GetSegmentDetection	635
GetTextDetection	640
IndexFaces	644
ListCollections	653
ListDatasetEntries	656
ListDatasetLabels	660
ListFaces	663
ListStreamProcessors	666
ListTagsForResource	669
RecognizeCelebrities	671
SearchFaces	676
SearchFacesByImage	680
StartCelebrityRecognition	685
StartContentModeration	689
StartFaceDetection	693
StartFaceSearch	697
StartLabelDetection	701
StartPersonTracking	705

StartProjectVersion	709
StartSegmentDetection	712
StartStreamProcessor	716
StartTextDetection	718
StopProjectVersion	722
StopStreamProcessor	724
TagResource	726
UntagResource	728
UpdateDatasetEntries	730
Tipos de datos	732
AgeRange	735
Asset	736
AudioMetadata	737
Beard	738
BlackFrame	739
BoundingBox	740
Celebrity	742
CelebrityDetail	744
CelebrityRecognition	746
ComparedFace	747
ComparedSourceImageFace	749
CompareFacesMatch	750
ContentModerationDetection	751
CoversBodyPart	752
CustomLabel	753
DatasetChanges	754
DatasetDescription	755
DatasetLabelDescription	757
DatasetLabelStats	758
DatasetMetadata	759
DatasetSource	761
DatasetStats	762
DetectionFilter	763
DetectTextFilters	764
DistributeDataset	765
Emotion	766
EquipmentDetection	767
EvaluationResult	768
Eyeglasses	769
EyeOpen	770
Face	771
FaceDetail	773
FaceDetection	776
FaceMatch	777
FaceRecord	778
FaceSearchSettings	779
Gender	780
Geometry	781
GroundTruthManifest	782
HumanLoopActivationOutput	783
HumanLoopConfig	784
HumanLoopDataAttributes	785
Image	786
ImageQuality	788
Instance	789
KinesisDataStream	790
KinesisVideoStream	791

KnownGender	792
Label	793
LabelDetection	794
Landmark	795
ModerationLabel	796
MouthOpen	797
Mustache	798
NotificationChannel	799
OutputConfig	800
Parent	801
PersonDetail	802
PersonDetection	803
PersonMatch	804
Point	805
Pose	806
ProjectDescription	807
ProjectVersionDescription	808
ProtectiveEquipmentBodyPart	811
ProtectiveEquipmentPerson	812
ProtectiveEquipmentSummarizationAttributes	813
ProtectiveEquipmentSummary	814
RegionOfInterest	816
S3Object	817
SegmentDetection	818
SegmentTypeInfo	821
ShotSegment	822
Smile	823
StartSegmentDetectionFilters	824
StartShotDetectionFilter	825
StartTechnicalCueDetectionFilter	826
StartTextDetectionFilters	827
StreamProcessor	828
StreamProcessorInput	829
StreamProcessorOutput	830
StreamProcessorSettings	831
Summary	832
Sunglasses	833
TechnicalCueSegment	834
TestingData	835
TestingDataResult	836
TextDetection	837
TextDetectionResult	839
TrainingData	840
TrainingDataResult	841
UnindexedFace	842
ValidationData	843
Video	844
VideoMetadata	845
Directrices y cuotas	847
Regiones admitidas	847
Establecer cuotas	847
Imagen de Amazon Rekognition	847
Vídeo almacenado de Amazon Rekognition Video	848
Amazon Rekognition Video streaming de vídeo	848
Cuotas predeterminadas	848
Calcular el cambio de cuota de TPS	849
Prácticas recomendadas para cuotas TPS	849

Crear un caso para cambiar las cuotas TPS	849
Historial de documentos	851
Glosario de AWS	856
.....	dccclvii

¿Qué es Amazon Rekognition?

Amazon Rekognition facilita la incorporación del análisis de imagen y vídeo a sus aplicaciones. Proporcione una imagen o un vídeo a la API de Amazon Rekognition y el servicio podrá identificar objetos, personas, texto, escenas y actividades. Asimismo, puede detectar cualquier contenido inadecuado. Amazon Rekognition proporciona además funcionalidades de análisis, comparación y búsqueda de rostros altamente precisas. 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.

Amazon Rekognition se basa en la misma tecnología de aprendizaje profundo de eficacia demostrada y altamente escalable desarrollada por los científicos de visión informática de Amazon para analizar miles de imágenes y vídeos al día. No requiere experiencia en aprendizaje automático para utilizarlo. Amazon Rekognition incluye una API sencilla y fácil de usar que puede analizar rápidamente cualquier archivo de imagen o vídeo almacenado en Amazon S3. Amazon Rekognition está aprendiendo siempre a partir de nuevos datos y añadimos continuamente nuevas etiquetas y características de comparación facial al servicio. Para obtener más información, consulte la[Preguntas frecuentes sobre Amazon Rekognition](#).

Entre los casos de uso comunes de Amazon Rekognition se incluyen los siguientes:

- Librerías de imágenes y vídeos con capacidad de búsqueda: Amazon Rekognition permite realizar búsquedas en las imágenes y vídeos almacenados para que pueda detectar los objetos y las escenas que aparecen en ellos.
- Verificación de usuarios basada en cara: Amazon Rekognition permite que sus aplicaciones confirmen las identidades de los usuarios comparando su imagen en vivo con una imagen de referencia.
- Detección de equipos de protección personal

Amazon Rekognition detecta el equipo de protección personal (EPP), como cubiertas faciales, fundas para la cabeza y cubiertas de manos en las personas que aparecen en imágenes. Puede utilizar la detección de EPP cuando la seguridad es lo más importante para nosotros. Por ejemplo, industrias como la construcción, la fabricación, la sanidad, el procesamiento de alimentos, la logística y el comercio minorista. Con la detección de EPP, puede detectar automáticamente si una persona lleva un tipo específico de EPP. Puede utilizar los resultados de detección para enviar una notificación o para identificar lugares en los que se pueden mejorar las advertencias de seguridad o las prácticas de formación.

- Análisis demográfico y de sentimientos: Amazon Rekognition interpreta expresiones emocionales como felicidad, tristeza y sorpresa, y datos demográficos como el sexo en las imágenes faciales. Amazon Rekognition puede analizar imágenes y enviar los atributos de emoción y demográficos a Amazon Redshift para realizar informes periódicos sobre las tendencias, como en establecimientos comerciales y situaciones similares. Tenga en cuenta que las predicciones de expresiones emocionales se basan únicamente en la apariencia física del rostro de una persona. No es indicativo del estado emocional interno y Rekognition no debe utilizarse para tomar dicha decisión.
- Búsqueda facial: con Amazon Rekognition, puede buscar imágenes, vídeos almacenados y vídeos en streaming 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. La identificación de personas en función de su rostro es un proceso que consta de dos pasos principales en Amazon Rekognition:

1. Indexar las caras

2. Buscar las caras

- Detección de contenido no seguro: Amazon Rekognition puede detectar contenido violento y para adultos en imágenes y vídeos almacenados. Los desarrolladores pueden usar los metadatos devueltos para filtrar contenido inadecuado en función de las necesidades del negocio. Además de marcar una imagen en función de la presencia de contenido para adultos, la API también devuelve una lista jerárquica de etiquetas con puntuaciones de confianza. Estas 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); por ejemplo, sitios de citas y redes sociales, plataformas de intercambio de datos, blogs y foros, aplicaciones para niños, sitios de comercio electrónico y servicios de entretenimiento y de publicidad online.
- Reconocimiento de famosos: Amazon Rekognition puede reconocer famosos dentro de las imágenes y vídeos proporcionados. Amazon Rekognition puede reconocer miles de famosos en una serie de categorías, como, por ejemplo, política, deportes, negocios, entretenimiento y medios de comunicación.
- Detección de texto: Amazon Rekognition Text in Image le permite reconocer y extraer contenido de texto a partir de imágenes. Texto en imágenes es compatible con la mayoría de las fuentes, incluso las que son muy estilizadas. Es capaz de detectar texto y números en diferentes orientaciones, como los que se suelen utilizar en pancartas y carteles. En las aplicaciones para compartir imágenes y de redes sociales, puede usarlo para habilitar la búsqueda visual basada en un índice de imágenes que contienen las mismas palabras clave. En las aplicaciones multimedia y de entretenimiento, puede catalogar vídeos en función del texto pertinente que aparece en pantalla, como anuncios, noticias, resultados deportivos y subtítulos. Finalmente, en las aplicaciones de seguridad pública, puede identificar vehículos basándose en los números de matrícula de las imágenes tomadas por las cámaras situadas en la vía pública.
- Etiquetas personalizadas: con las etiquetas personalizadas de Amazon Rekognition, puede identificar los objetos y las escenas de las imágenes específicas de sus necesidades empresariales. Por ejemplo, puede encontrar su logotipo en publicaciones de redes sociales, identificar sus productos en los estantes de las tiendas, clasificar piezas de máquinas en una línea de montaje, distinguir plantas en buen estado o infectadas, o detectar sus personajes animados en vídeos. Para obtener más información, consulte [¿Qué son las etiquetas personalizadas de Amazon Rekognition?](#) en la Guía del desarrollador de etiquetas personalizadas de Amazon Rekognition.

Alguno de los beneficios de usar Amazon Rekognition son:

- Integración de potentes análisis de imagen y vídeo en sus aplicaciones: no necesita conocimientos en visión artificial o aprendizaje profundo para usar el análisis de imágenes y vídeo de confianza de Amazon Rekognition. Con la API, puede realizar análisis de imágenes y vídeo de manera fácil y rápida en cualquier aplicación web o móvil, así como en cualquier dispositivo conectado.
- Análisis de imágenes y vídeo basado en el aprendizaje profundo: Amazon Rekognition utiliza la tecnología de aprendizaje profundo para analizar las imágenes con exactitud, buscar y comparar rostros en imágenes y detectar objetos y escenas en sus imágenes y vídeos.
- Análisis de imágenes escalable: Amazon Rekognition le permite analizar millones de imágenes, por lo que puede reunir y organizar grandes cantidades de datos visuales.

- **Integración de con otros servicios de AWS:** Amazon Rekognition se ha diseñado para funcionar perfectamente con otros servicios de AWS como Amazon S3 y AWS Lambda. Puede llamar a la API de Amazon Rekognition directamente desde Lambda en respuesta a eventos de Amazon S3. Como Amazon S3 y Lambda se amplían automáticamente en función de la demanda de su aplicación, puede crear aplicaciones de análisis de imágenes escalables, asequibles y fiables. Por ejemplo, cada vez que una persona llegue a su residencia, la cámara de la puerta puede cargar una foto del visitante en Amazon S3. Esto activará una función de Lambda que utiliza operaciones de API de Amazon Rekognition para identificar al invitado. Puede ejecutar análisis directamente en las imágenes almacenadas en Amazon S3 sin tener que cargar o mover los datos. Compatibilidad con AWS Identity and Access Management (IAM) facilita la tarea de controlar de forma segura el acceso a las operaciones de API de Amazon Rekognition. Con IAM, puede crear y administrar usuarios y grupos de AWS para conceder el acceso apropiado a sus desarrolladores y usuarios finales.
- **Bajo costo:** con Amazon Rekognition, paga por el número de imágenes y videos que analiza y los metadatos de rostros que almacena. No se requieren pagos mínimos ni compromisos iniciales. Puede empezar con la tarifa gratuita y disfrutar de un gran ahorro cuando se amplíen sus necesidades gracias al modelo de precios por niveles de Amazon Rekognition.

Elegibilidad de Amazon Rekognition y 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 es un usuario nuevo de Amazon Rekognition, le recomendamos que lea las siguientes secciones en orden:

1. [Cómo funciona Amazon Rekognition \(p. 4\)](#): esta sección presenta varios componentes de Amazon Rekognition con los que va a trabajar para crear una experiencia integral.
2. [Introducción a Amazon Rekognition \(p. 12\)](#): en esta sección va a configurar su cuenta y probar la API de Amazon Rekognition.
3. [Trabajar con imágenes \(p. 28\)](#): 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. [Trabajar con vídeos almacenados \(p. 63\)](#): esta sección proporciona información sobre el uso de Amazon Rekognition con vídeos almacenados en un bucket de Amazon S3.
5. [Uso de vídeos en streaming \(p. 94\)](#): 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. Usted utiliza Amazon Rekognition Image para analizar imágenes, y Amazon Rekognition Video para analizar vídeos.

Las dos API analizan imágenes y vídeos para proporcionar información que después puede utilizar en sus aplicaciones. Por ejemplo, podría utilizar Amazon Rekognition Image para mejorar la experiencia de los clientes para una aplicación de administración de fotos. Cuando un cliente carga una foto, la aplicación puede utilizar Amazon Rekognition Image para detectar objetos del mundo real o rostros en la imagen. Después de que la aplicación almacene la información que devuelve Amazon Rekognition Image, el usuario podría consultar su colección de fotos para localizar fotos con un objeto o rostro concreto. Es posible una consulta más profunda. Por ejemplo, el usuario podría realizar consultas de rostros que sonrían o consultar rostros de una determinada edad.

Puede utilizar Amazon Rekognition Video para realizar un seguimiento del recorrido de las personas en un vídeo almacenado. De forma alternativa, puede utilizar Amazon Rekognition Video para buscar un vídeo en streaming para localizar personas cuyas descripciones faciales coincidan con las descripciones faciales ya almacenadas por Amazon Rekognition.

La API de Amazon Rekognition realiza un análisis de imágenes de aprendizaje profundo fácil de usar. Por ejemplo, [RecognizeCelebrities \(p. 671\)](#) devuelve información para un máximo de 100 famosos detectados en una imagen. Esto incluye información acerca de dónde se han detectado los rostros de famosos en la imagen y dónde obtener información adicional acerca del famoso.

La siguiente información trata 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.

Temas

- [Tipos de análisis \(p. 4\)](#)
- [Operaciones de imágenes y vídeo \(p. 6\)](#)
- [Operaciones de API sin almacenamiento y almacenamiento \(p. 8\)](#)
- [Control de versiones del modelo \(p. 10\)](#)

Tipos de análisis

Estos son los tipos de análisis que la API de Image de Amazon Rekognition y la API de Amazon Rekognition Video pueden realizar. Para obtener más información acerca de los API, consulte [Operaciones de imágenes y vídeo \(p. 6\)](#).

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, salir de un vehículo). Amazon Rekognition puede detectar etiquetas en imágenes y vídeos. Sin embargo, las actividades no se detectan en las imágenes. Para obtener más información, consulte [Detección de etiquetas \(p. 135\)](#).

Para detectar etiquetas en imágenes, utilice [DetectLabels \(p. 583\)](#). Para detectar etiquetas en vídeos almacenados, utilice [StartLabelDetection \(p. 701\)](#).

Etiquetas personalizadas

Las etiquetas personalizadas de Amazon Rekognition pueden identificar los objetos y las escenas de las imágenes específicas de sus necesidades empresariales entrenando a un modelo de aprendizaje automático. 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 información sobre las 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 aprendizaje automático. Para obtener más información, consulte [Introducción a las 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 y ejecutar un modelo. 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 con un modelo entrenado, utilice [DetectCustomLabels](#).

Rostros

Amazon Rekognition puede detectar rostros en imágenes y vídeos almacenados. Con Amazon Rekognition puede obtener información acerca de dónde se detectan rostros en una imagen o vídeo, referencias faciales como, por ejemplo, la posición de los ojos y las emociones detectadas como felicidad o tristeza. También 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 \(p. 149\)](#).

Para detectar rostros en imágenes, utilice [DetectFaces \(p. 578\)](#). Para detectar rostros en vídeos almacenados, utilice [StartFaceDetection \(p. 693\)](#).

Búsqueda de rostros

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, consulte [Búsqueda de rostros en una colección \(p. 181\)](#).

Para buscar rostros conocidos en imágenes, utilice [DetectFaces \(p. 578\)](#). Para buscar rostros conocidos en vídeos almacenados, utilice [StartFaceDetection \(p. 693\)](#). Para buscar rostros conocidos en vídeos en streaming, utilice [CreateStreamProcessor \(p. 537\)](#).

Recorridos del personal

Amazon Rekognition puede realizar un seguimiento de las rutas de las personas detectadas en un vídeo almacenado. Amazon Rekognition Video proporciona información de seguimiento de rutas, de detalles de rostros y de ubicación en fotograma de las personas detectadas en un vídeo. Para obtener más información, consulte [Recorridos de las personas \(p. 243\)](#).

Para detectar personas en vídeos almacenados, utilice [StartPersonTracking \(p. 705\)](#).

Equipo de protección personal

Amazon Rekognition puede detectar el equipo de protección personal (EPP) usado por las personas detectadas en una imagen. Amazon Rekognition detecta cubiertas faciales, fundas de manos y cubiertas para la cabeza. Amazon Rekognition predice si un elemento de EPP cubre la parte del cuerpo adecuada. También puede obtener cuadros delimitadores para personas detectadas y elementos de EPP. Para obtener más información, consulte [Detección de equipos de protección personal \(p. 251\)](#).

Para detectar EPP en imágenes, utilice [DetectProtectiveEquipment \(p. 592\)](#).

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 información adicional sobre un famoso reconocido, como la emoción expresada y la presentación de género. Para obtener más información, consulte [Reconocimiento de famosos \(p. 275\)](#).

Para reconocer famosos en imágenes, utilice [RecognizeCelebrities \(p. 671\)](#). Para reconocer famosos en vídeos almacenados, utilice [StartCelebrityRecognition \(p. 685\)](#).

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 \(p. 315\)](#).

Para detectar texto en imágenes, utilice [DetectText \(p. 596\)](#).

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 de contenido \(p. 298\)](#).

Para detectar imágenes no seguras, utilice [DetectModerationLabels \(p. 588\)](#). Para detectar vídeos almacenados no seguros, utilice [StartContentModeration \(p. 689\)](#).

Operaciones de imágenes y vídeo

Amazon Rekognition proporciona dos conjuntos de API. Usted utiliza Amazon Rekognition Image para analizar imágenes, y Amazon Rekognition Video para analizar vídeos almacenados y en streaming. El siguiente tema ofrece una breve información general de cada conjunto de API.

Amazon Rekognition Image y Amazon Rekognition Video API de pueden detectar diferentes entidades, como rostros u objetos. Para obtener información sobre los tipos de comparación y detección admitidos, consulte [Tipos de análisis \(p. 4\)](#).

Operaciones de Amazon Rekognition Image

Las operaciones de imágenes de Amazon Rekognition son síncronas. 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 pasa a una operación de Amazon Rekognition Image que puede estar almacenada en un bucket de Amazon S3. Si no va a utilizar la AWS

CLI, también puede transferir bytes de imágenes codificados en Base64 directamente a una operación de Amazon Rekognition. Para obtener más información, consulte [Trabajar con imágenes \(p. 28\)](#).

Operaciones de Amazon Rekognition Video

Amazon Rekognition Video puede analizar vídeos almacenados en un bucket de Amazon S3 y vídeos en streaming a través de Amazon Kinesis Video Streams.

Las operaciones de vídeo de Amazon Rekognition Video son asíncronas. Con las operaciones de vídeo de almacenamiento de Amazon Rekognition Video, inicia el análisis llamando a la operación de inicio para el tipo de análisis que desee. Por ejemplo, para detectar rostros en un vídeo almacenado, llame a [StartFaceDetection \(p. 693\)](#). Una vez completado, Amazon Rekognition publica el estado de realización en un tema de Amazon SNS. Para obtener los resultados de la operación de análisis, llame a la operación de obtención para el tipo de análisis que ha solicitado; por ejemplo, [GetFaceDetection \(p. 615\)](#). Para obtener más información, consulte [Trabajar con vídeos almacenados \(p. 63\)](#).

Con las operaciones de vídeo de streaming de Amazon Rekognition Video, puede buscar rostros almacenados en colecciones de Amazon Rekognition Video. Amazon Rekognition Video analiza un flujo de vídeo de Kinesis y publica la salida de los resultados de búsqueda en un flujo de datos de Kinesis. El análisis de vídeo se administra mediante la creación y uso de un procesador de streaming de Amazon Rekognition Video. Por ejemplo, crea un procesador de streaming llamando a [CreateStreamProcessor \(p. 537\)](#). Para obtener más información, consulte [Uso de vídeos en streaming \(p. 94\)](#).

Operaciones sin almacenamiento de información y basadas en almacenamiento de información

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

- Operaciones de API sin almacenamiento de información— 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 \(p. 8\)](#).
- Operaciones de API basadas en almacenamiento de información: 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 basadas en almacenamiento de información \(p. 9\)](#).

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

Puede llamar a las operaciones de API de Amazon Rekognition utilizando el SDK de AWS 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 for .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.

La [Referencia de la API \(p. 511\)](#)En esta guía se explica cómo llamar a las operaciones de Amazon Rekognition utilizando HTTP. Para información de referencia de Java, consulte [AWS SDK for Java](#).

Los puntos de enlace del servicio de Amazon Rekognition que puede utilizar se documentan en[Regiones y puntos de enlace de AWS](#).

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

Operaciones de API sin almacenamiento y 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 \(p. 583\)](#)
- [DetectFaces \(p. 578\)](#)
- [CompareFaces \(p. 515\)](#)
- [DetectModerationLabels \(p. 588\)](#)
- [DetectProtectiveEquipment \(p. 592\)](#)
- [RecognizeCelebrities \(p. 671\)](#)
- [DetectText \(p. 596\)](#)
- [GetCelebrityInfo \(p. 602\)](#)

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

- [StartLabelDetection \(p. 701\)](#)
- [StartFaceDetection \(p. 693\)](#)
- [StartPersonTracking \(p. 705\)](#)
- [StartCelebrityRecognition \(p. 685\)](#)
- [StartContentModeration \(p. 689\)](#)

Estos elementos se denominansin almacenamientoOperaciones de API 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 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 etiquetas utilizando Amazon Rekognition[DetectLabels](#)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 el `Amazon RekognitionDetectFaces` operación de en el cliente y crea 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 basadas en almacenamiento de información

Amazon Rekognition Image admite el [IndexFaces](#) (p. 644), 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](#) (p. 644)
- [ListFaces](#) (p. 663)
- [SearchFacesByImage](#) (p. 680)
- [SearchFaces](#) (p. 676)

- [DeleteFaces \(p. 545\)](#)
- [DescribeCollection \(p. 556\)](#)
- [DeleteCollection \(p. 541\)](#)
- [ListCollections \(p. 653\)](#)
- [CreateCollection \(p. 522\)](#)

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

- [StartFaceSearch \(p. 697\)](#)
- [CreateStreamProcessor \(p. 537\)](#)

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`.

Información facial almacenada en colecciones por `IndexFaces` está accesible 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 \(p. 697\)](#).

Para obtener más información acerca de cómo se crean y se administran colecciones, consulte [Búsqueda de rostros en una colección \(p. 181\)](#).

Note

El servicio no conserva bytes de imágenes reales. En lugar de ello, el algoritmo de detección subyacente detecta primero los rostros en la imagen de entrada, extrae los rasgos faciales en un vector de rasgos de cada rostro y después los almacena en la base de datos. Amazon Rekognition usa estos vectores de rasgos cuando busca rostros coincidentes.

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.

Control de versiones del modelo

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 \(p. 644\)](#) 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 \(p. 644\)](#) 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 \(p. 556\)](#). Para obtener más información, consulte [Descripción de una colección \(p. 196\)](#).

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 \(p. 522\)](#)) 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 \(p. 541\)](#).

Las operaciones sin estado, como [DetectFaces \(p. 578\)](#), 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 \(p. 4\)](#).

Antes de utilizar Rekognition, es preciso crear una cuenta de AWS y obtener un ID de cuenta de AWS. También querrá crear un usuario de IAM, lo que permite al sistema Amazon Rekognition determinar si tiene los permisos necesarios para acceder a sus recursos.

Después de crear las cuentas, querrás instalar y configurar el AWS CLIyAWSSDK. La AWS CLI te permite interactuar con Amazon Rekognition y otros servicios a través de la línea de comandos, mientras que AWS SDK te permiten utilizar lenguajes de programación como Java y Python para interactuar con Amazon Rekognition.

Una vez configurados la AWS CLIyAWSSDK, puedes ver algunos ejemplos de cómo usarlos ambos. También puedes ver algunos ejemplos de cómo interactuar con Amazon Rekognition mediante la consola.

Temas

- [Paso 1: Configurar una cuenta de AWS y crear un usuario de IAM \(p. 12\)](#)
- [Paso 2: Configurar la AWS CLIyAWSSDK de \(p. 14\)](#)
- [Paso 3: Primeros pasos con AWS CLIyAWSSDK API \(p. 15\)](#)
- [Paso 4: Primeros pasos con la consola de Amazon Rekognition \(p. 16\)](#)

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

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

1. [Inscripción en AWS \(p. 12\)](#)
2. [Creación de un usuario de IAM \(p. 13\)](#)

Inscripción en 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 cliente nuevo de AWS, puede comenzar a utilizar Amazon Rekognition gratuitamente. Para obtener más información, consulte [Capa gratuita de AWS](#).

Si ya dispone de una cuenta de AWS, pase a la siguiente tarea. Si todavía no dispone de una cuenta de AWS, realice los pasos del siguiente procedimiento para crear una.

Para crear una cuenta de AWS

1. Abra <https://portal.aws.amazon.com/billing/signup>.
2. Siga las instrucciones en línea.

Parte del procedimiento de inscripción consiste en recibir una llamada telefónica e indicar un código de verificación en el teclado del teléfono.

Anote su ID de cuenta de AWS porque lo necesitará en la siguiente tarea.

Creación de un usuario de IAM

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. La consola requiere que especifique la contraseña. Puede crear claves de acceso para su cuenta de AWS para tener acceso a la AWS CLI o API. Sin embargo, no es recomendable que acceda a AWS con las credenciales de usuario raíz de su cuenta de AWS. Le recomendamos que utilice en su lugar:

- UsarAWS Identity and Access Management(IAM) para crear un usuario de IAM.
- Añada el usuario a un grupo de IAM con permisos administrativos.

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

Si se ha inscrito en AWS, pero no ha creado un usuario de IAM para usted, puede crear uno mediante la consola de IAM. Siga el procedimiento para crear un usuario de IAM en su cuenta.

Para crear un usuario de IAM e iniciar sesión en la consola

1. Cree un usuario de IAM con permisos de administrador en su cuenta de AWS. Para obtener instrucciones, consulte[Creación del primer grupo de usuarios y administradores de IAM](#)en la[IAM User Guide](#).
2. Como usuario de IAM, puede iniciar sesión en la AWS Management Console con una URL especial. Para obtener más información, consulte[Cómo inician sesión los usuarios en la cuenta](#)en la[IAM User Guide](#).

Note

Un usuario de IAM con permisos de administrador tiene acceso ilimitado a laAWSservicios de tu cuenta. Para obtener más información acerca de cómo restringir el acceso a las operaciones de Amazon Rekognition, consulte[Políticas de Amazon Rekognition basadas en identidades \(p. 486\)](#). En los ejemplos de código de esta guía se presupone que tiene un usuario con laAmazonRekognitionFullAccesspermisos. AmazonS3ReadOnlyAccessse requiere para ver ejemplos que tienen acceso a imágenes o vídeos que se almacenan en un bucket de Amazon S3. Los ejemplos de código de vídeo almacenado de Amazon Rekognition Video también requierenAmazonSQSFullAccesspermisos. En función de sus requisitos de seguridad, es posible que desee utilizar un grupo de IAM que se limite a estos permisos. Para obtener más información, consulte[Creación de un grupo de IAM](#).

Para obtener más información sobre IAM, consulte lo siguiente:

- [AWS Identity and Access Management \(IAM\)](#)
- [Introducción](#)
- [Guía del usuario de IAM](#)

Paso siguiente

[Paso 2: Configurar laAWS CLl y AWSSDK de \(p. 14\)](#)

Paso 2: Configurar laAWS CLIyAWSSDK de

En los pasos siguientes se muestra cómo instalar la AWS Command Line Interface (AWS CLI) y los AWS SDK que utilizan los ejemplos de esta documentación. Hay una serie de diferentes maneras para autenticar las llamadas al AWS SDK. En los ejemplos que aparecen en esta guía se presupone que utiliza un perfil de credenciales predeterminado para llamar a los comandos de la AWS CLI y las operaciones de API de AWS SDK.

Para ver la lista de disponiblesAWSRegiones, consulte[Regiones y puntos de enlace deen laReferencia general de Amazon Web Services](#).

Siga los pasos para descargar y configurar los AWS SDK.

Para configurar la AWS CLI y los AWS SDK

1. Descargue e instale la [AWS CLI](#) y los SDK de AWS que desea utilizar. En esta guía, se proporcionan ejemplos de la AWS CLI, Java, Python, Ruby, Node.js, PHP, .NET y JavaScript. Para obtener información sobre la instalaciónAWSSDK, consulte[Herramientas para Amazon Web Services](#).
2. Cree una clave de acceso para el usuario que ha creado en [Creación de un usuario de IAM \(p. 13\)](#).
 - a. Inicie sesión en la AWS Management Console y abra la consola de IAM en <https://console.aws.amazon.com/iam/>.
 - b. En el panel de navegación, seleccione Users.
 - c. Elija el nombre del usuario que ha cread en [Creación de un usuario de IAM \(p. 13\)](#).
 - 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 laAWS CLI, puede[configurar las credenciales y la región para la mayoríaAWSSDK introduciendoaws configureen el símbolo del sistema](#). 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
```

Sustituya su ID de clave de acceso y la clave de acceso secreta por`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
```

Sustituya la región de AWS deseada (por ejemplo, us-west-2) para tu aws_region.

Note

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

10. Guarde el archivo config.

Paso siguiente

[Paso 3: Primeros pasos con AWS CLI y AWSSDK API \(p. 15\)](#)

Uso de Rekognition con un AWSSDK

Los kits de desarrollo de software (SDK) de AWS están disponibles en 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 for C++	Ejemplos de código de AWS SDK for C++
AWS SDK for Go	Ejemplos de código de AWS SDK for Go
AWS SDK for Java	Ejemplos de código de AWS SDK for Java
AWS SDK for JavaScript	Ejemplos de código de AWS SDK for JavaScript
AWS SDK for .NET	Ejemplos de código de AWS SDK for .NET
AWS SDK for PHP	Ejemplos de código de AWS SDK for PHP
AWS SDK for Python (Boto3)	Ejemplos de código de AWS SDK for Python (Boto3)
AWS SDK for Ruby	Ejemplos de código de AWS SDK for Ruby

Ejemplo de disponibilidad

¿No puede encontrar lo que necesita? Solicite un ejemplo de código mediante el enlace [Provide feedback \(Enviar comentarios\)](#) en la parte inferior de esta página.

Paso 3: Primeros pasos con AWS CLI y AWSSDK API

Después de configurar el AWS CLI y AWS Para usar, puede compilar aplicaciones que usen Amazon Rekognition. En los siguientes temas, se muestra cómo comenzar a utilizar Amazon Rekognition Image y Amazon Rekognition Video.

- Trabajar con imágenes (p. 28)
- Trabajar con vídeos almacenados (p. 63)
- Uso de vídeos en streaming (p. 94)

Formato del AWS CLI Ejemplos de

Los ejemplos de AWS CLI incluidos en esta guía tienen formato 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](#). A continuación se muestra un ejemplo de comando de AWS CLI formateado para Microsoft Windows.

```
awsrekognition detect-labels ^
--image "{\"S3Object\":{\"Bucket\":\"photo-collection\",\"Name\":\"photo.jpg\"}}" ^
--region us-west-2
```

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

```
awsrekognition detect-labels --image "S3Object={Bucket=photo-collection,Name=photo.jpg}"
--region us-west-2
```

Para obtener más información, consulte [Uso de sintaxis abreviada con la AWS Command Line Interface](#).

Paso siguiente

[Paso 4: Primeros pasos con la consola de Amazon Rekognition \(p. 16\)](#)

Paso 4: Primeros pasos con la consola de Amazon 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 \(p. 4\)](#). También puede utilizar la API de Amazon Rekognition o AWS CLI para detectar objetos y escenas, detectar rostros y comparar y buscar rostros. Para obtener más información, consulte [Paso 3: Primeros pasos con AWS CLI y AWSSDK API \(p. 15\)](#).

En esta sección se indica también cómo ver métricas globales de Amazon CloudWatch para Rekognition mediante la consola de Rekognition.

Temas

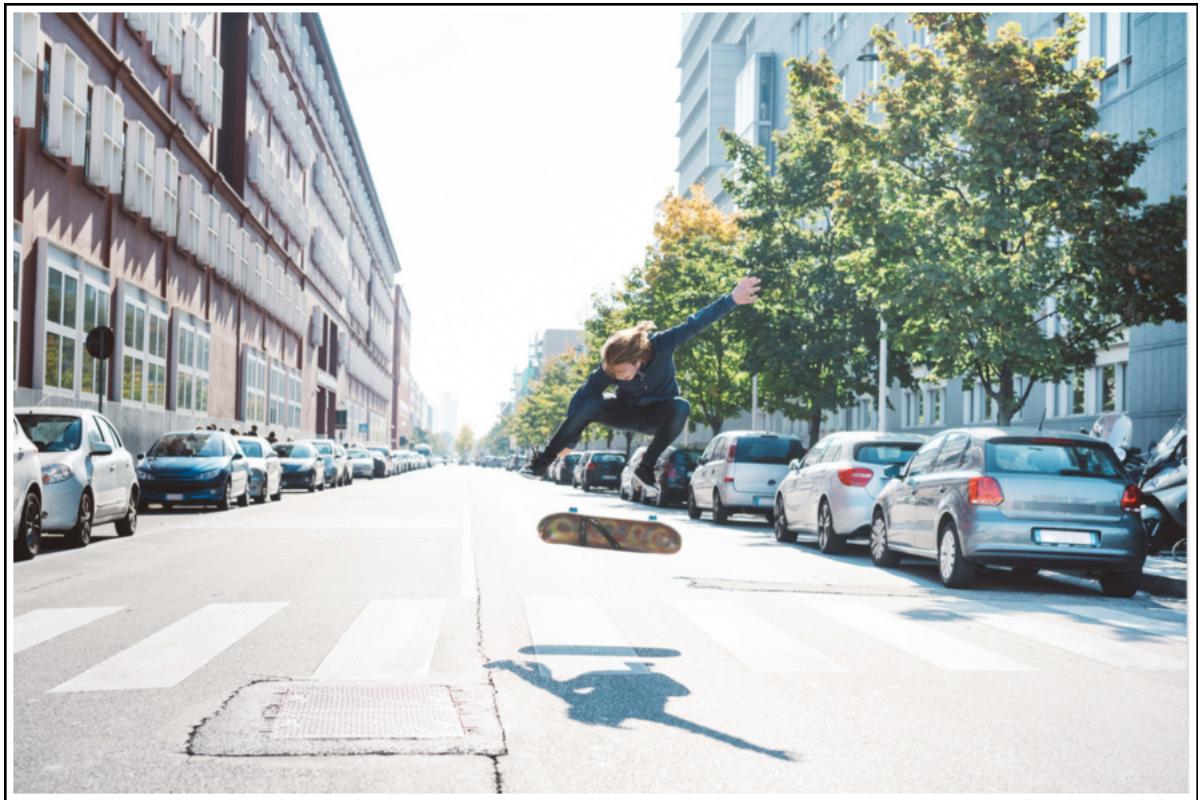
- [Ejercicio 1: Detección de objetos y escenas \(consola\) \(p. 17\)](#)
- [Ejercicio 2: Analizar rostros en una imagen \(consola\) \(p. 22\)](#)
- [Ejercicio 3: Comparar caras en imágenes \(consola\) \(p. 24\)](#)
- [Ejercicio 4: Ver métricas globales \(consola\) \(p. 26\)](#)



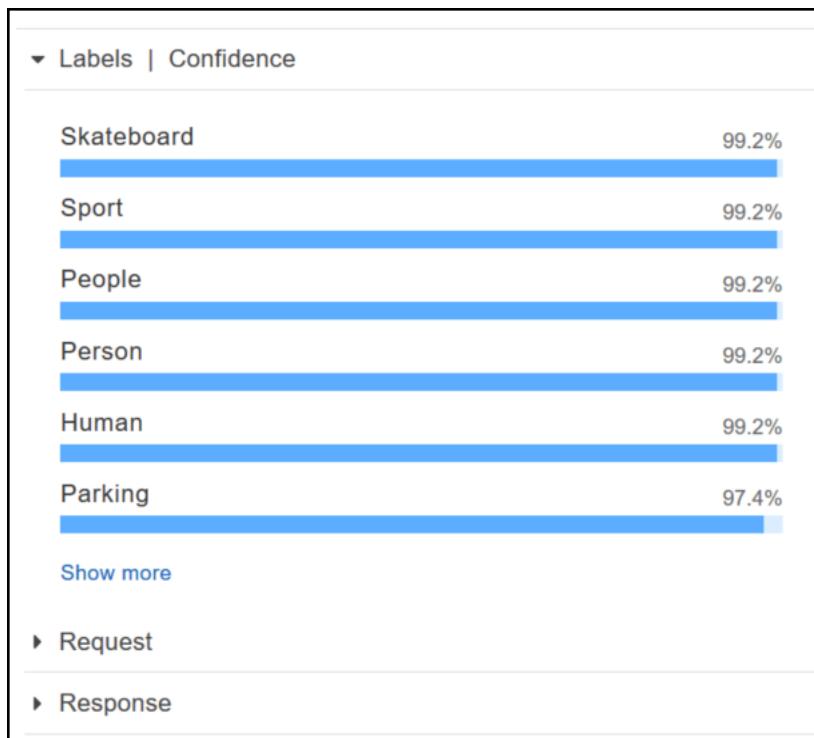
Ejercicio 1: Detección de objetos y escenas (consola)

Esta sección muestra cómo funciona, a un nivel muy alto, 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":99.25359344482422,  
            "Name": "People"  
        },  
        {  
            "Confidence":99.25359344482422,  
            "Name": "Person"  
        },  
        {  
            "Confidence":99.25359344482422,  
            "Name": "Human"  
        },  
        {  
            "Confidence":99.25359344482422,  
            "Name": "Sport"  
        },  
        {  
            "Confidence":99.25359344482422,  
            "Name": "Skateboard"  
        }  
    ]  
}
```

```
{  
    "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 \(p. 4\)](#).

Detectar objetos y escenas en una imagen proporcionada por el usuario

Puede cargar 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, 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 de Amazon Rekognition en<https://console.aws.amazon.com/rekognition/>.
2. ElegirDetección de etiqueta.
3. Realice una de las siguientes acciones:
 - Cargue 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” \(p. 28\)](#).

Detecta objetos y personas en un vídeo que proporcionas

Puede cargar un vídeo que proporcione 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 durar más de un minuto ni superar los 30 MB. Debe estar en formato de archivo MP4 y codificado con el códec H.264.

Para detectar objetos y personas en un vídeo proporcionado por el usuario

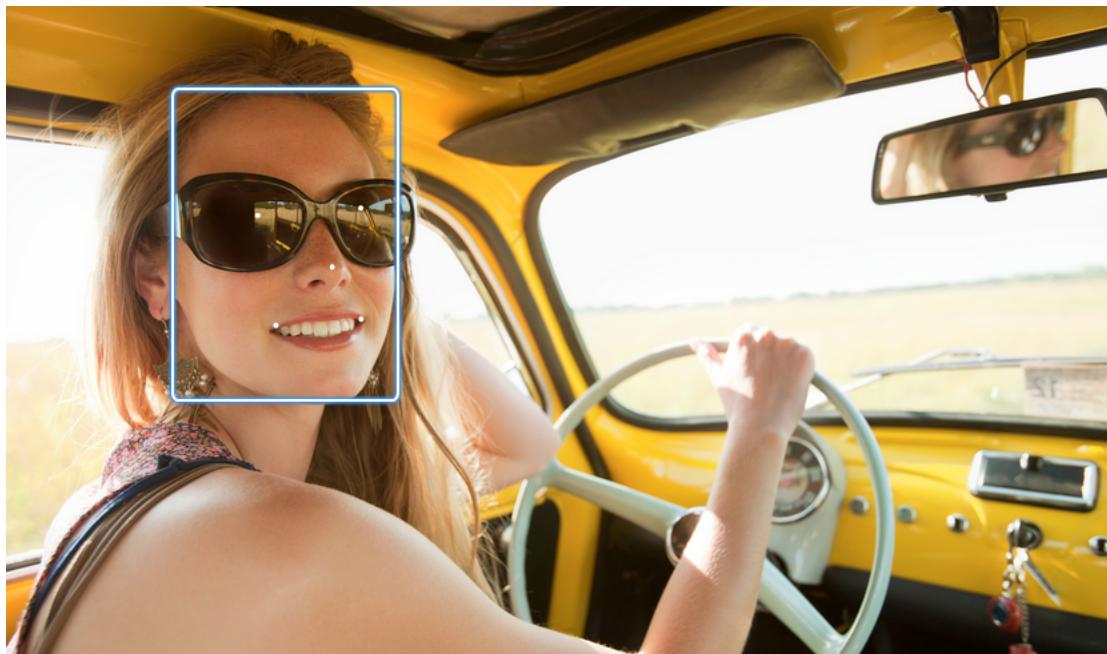
1. Abra la consola de Amazon Rekognition en<https://console.aws.amazon.com/rekognition/>.
2. ElegirAnálisis de vídeo.
3. BajoElige una muestra o carga la tuya propia, seleccioneSu propio vídeo.
4. Arrastra y suelta el vídeo o selecciona el vídeo desde la ubicación en la que lo has almacenado.

Para obtener más opciones de análisis de vídeo, consulte [the section called “Trabajar con vídeos almacenados” \(p. 63\)](#) o [the section called “Uso de vídeos en streaming” \(p. 94\)](#).

Ejercicio 2: Analizar rostros en 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 \(p. 4\)](#).

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 la acciónRostros | Confianzapanel.

Analizar rostros en una imagen proporcionada por el usuario

Puede cargar 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 de Amazon Rekognition en <https://console.aws.amazon.com/rekognition/>.

2. Elija Facial analysis.
3. Realice una de las siguientes acciones:
 - Cargue 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 caras en imágenes (consola)

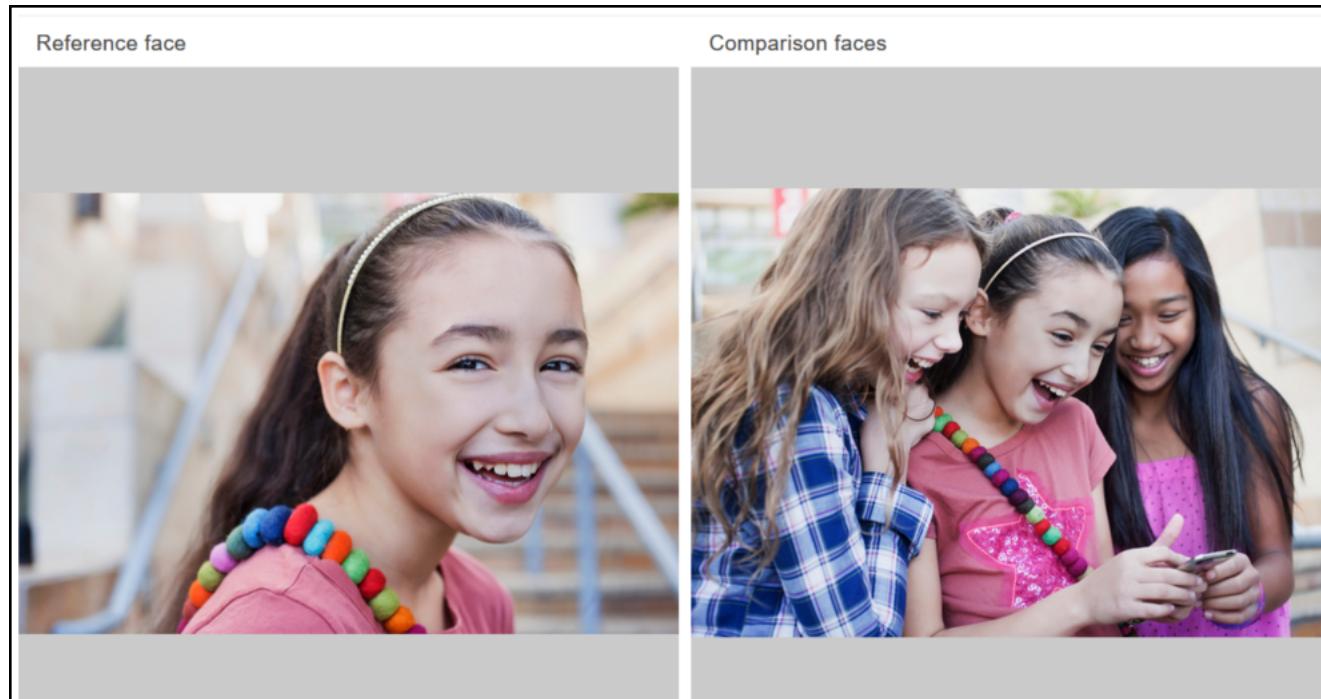
Esta sección muestra cómo utilizar la consola de Amazon Rekognition para comparar rostros en un conjunto de imágenes con varios rostros en ellas. Cuando especifica unFace de referencia(fuente) y unRostros de comparaciónImagen (destino), Rekognition compara el rostro más grande de la imagen de origen (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.

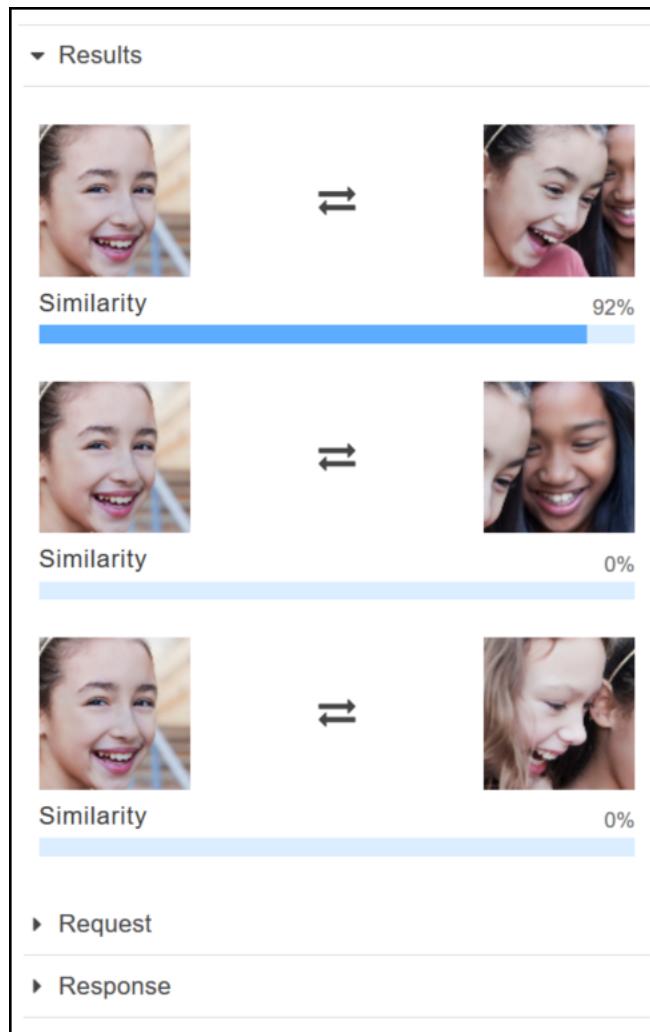
Si la imagen de origen contiene varios rostros, el servicio detecta el rostro más grande en la imagen de origen y lo usa para compararlo con cada rostro detectado en la imagen de destino.

Para obtener más información, consulte [Comparación de rostros en imágenes \(p. 163\)](#).

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 en una imagen proporcionada por el usuario

Puede cargar 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 de 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:
 - Cargue 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:
 - Cargue 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 la imagen de destinoResultadospanel.

Ejercicio 4: Ver métricas globales (consola)

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

En la siguiente tabla se indican los gráficos que se muestran en el panel de métricas de Rekognition y la métrica de Rekognition correspondiente. Para obtener más información, consulte [Métricas de CloudWatch para Rekognition \(p. 503\)](#).

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 API individuales, elija el enlace situado debajo de cada gráfico.

Para permitir a los usuarios acceso al panel de métricas de Rekognition, asegúrese de que el usuario tiene los permisos de CloudWatch y Rekognition adecuados. 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 [Identity and Access Management para Amazon Rekognition \(p. 481\)](#).

Para obtener más información sobre la monitorización de Rekognition con CloudWatch, consulte [Monitorización Rekognition \(p. 500\)](#).

Para ver métricas globales (consola)

1. Abra la consola de Amazon Rekognition en<https://console.aws.amazon.com/rekognition/>.
2. En el panel de navegación, seleccione Metrics (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 métricas de CloudWatch detalladas para una métrica global específica, elija Ver detalles sobre CloudWatch debajo del gráfico métrico.

Trabajo con imágenes y vídeos

Puede utilizar las operaciones de la API de Amazon Rekognition con imágenes, vídeos almacenados y vídeos en streaming. En esta sección se proporciona información general sobre cómo escribir código que obtiene acceso a Amazon Rekognition. 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 \(p. 28\)](#)
- [Trabajar con vídeos almacenados \(p. 63\)](#)
- [Uso de vídeos en streaming \(p. 94\)](#)
- [Control de errores \(p. 122\)](#)
- [Uso de Amazon Rekognition como servicio autorizado de FedRAMP \(p. 127\)](#)

Trabajar con imágenes

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

- [Detección de objetos y escenas \(p. 135\)](#)
- [Detección y comparación de rostros \(p. 149\)](#)
- [Búsqueda de rostros en una colección \(p. 181\)](#)
- [Reconocimiento de famosos \(p. 275\)](#)
- [Moderación de imágenes \(p. 298\)](#)
- [Detección de texto en una imagen \(p. 315\)](#)

Se llevan a cabo mediante operaciones 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 [Operaciones de API sin almacenamiento y almacenamiento \(p. 8\)](#).

Amazon Rekognition Image también puede almacenar metadatos faciales en colecciones para recuperación posterior. Para obtener más información, consulte [Búsqueda de rostros en una colección \(p. 181\)](#).

En esta sección se utilizan las operaciones API de Amazon Rekognition Image API para analizar imágenes almacenadas en un bucket de Amazon S3 y bytes de imagen cargados 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.

Temas

- [Especificaciones de imágenes \(p. 29\)](#)
- [Análisis de imágenes almacenadas en un bucket de Amazon S3 \(p. 30\)](#)
- [Análisis de una imagen cargada desde un sistema de archivos local \(p. 39\)](#)
- [Visualización de cuadros delimitadores \(p. 49\)](#)
- [Obtener orientación de imagen y coordenadas de cuadro delimitador \(p. 57\)](#)

Especificaciones de imágenes

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 Amazon 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 \(p. 30\)](#). Para ver un ejemplo del paso de bytes de imagen a una operación API de imagen de Amazon Rekognition, consulte [Análisis de una imagen cargada desde un sistema de archivos local \(p. 39\)](#).

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.

El siguiente comúne AWS Los SDK cifran las imágenes en base64 automáticamente y no es necesario cifrar bytes de imagen antes de llamar a una operación API de Amazon Rekognition Image API de.

- 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 utiliza el AWS CLI Para llamar a las operaciones de Amazon Rekognition Image, no es posible transferir bytes de imágenes como parte de la llamada. Debe cargar primero la imagen en un bucket de Amazon S3 y, a continuación, llamar a la operación que hace referencia a la imagen cargada.

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 imagen de Amazon Rekognition \(p. 131\)](#).

Corrección de 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 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 [Obtener orientación de imagen y coordenadas de cuadro delimitador \(p. 57\)](#).

Redimensionamiento de imágenes

Durante el análisis, Amazon Rekognition cambia el tamaño interno de las imágenes utilizando un conjunto de rangos predefinidos que mejor se adaptan a un modelo o algoritmo en particular. Debido a esto, Amazon Rekognition puede detectar un número diferente de objetos o proporcionar resultados diferentes, según la resolución de la imagen de entrada. Supongamos que tiene dos imágenes. La primera imagen

tiene una resolución de 1024x768 píxeles. La segunda imagen, una versión redimensionada de la primera imagen, tiene una resolución de 640x480 píxeles. Si envía las imágenes a [the section called "DetectLabels" \(p. 583\)](#), las respuestas de las dos llamadas a `DetectLabels` podrían 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 el [DetectLabels \(p. 583\)](#) Operación API 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 de Amazon Rekognition Image API de mediante el [the section called "Image" \(p. 786\)](#) parámetro de entrada. Dentro de `Image`, especifique la propiedad de objeto [S3Object \(p. 817\)](#) 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 codificadas en base64. Para obtener más información, consulte [Especificaciones de imágenes \(p. 29\)](#).

Ejemplo de solicitud

En este ejemplo, solicitud JSON para `DetectLabels`, la imagen de origen (`input.jpg`) se carga desde un bucket de Amazon S3 denominado `MyBucket`. Tenga en cuenta que la región del bucket de S3 que contiene el objeto S3 debe coincidir con la región que utiliza para las operaciones de Amazon Rekognition Image.

```
{  
    "Image": {  
        "S3Object": {  
            "Bucket": "MyBucket",  
            "Name": "input.jpg"  
        }  
    },  
    "MaxLabels": 10,  
    "MinConfidence": 75  
}
```

En los siguientes ejemplos se emplean varios AWS SDK y la AWS CLI para llamar a `DetectLabels`. Para obtener más información sobre la respuesta de la operación `DetectLabels`, consulte [Respuesta DetectLabels \(p. 143\)](#).

Para detectar las etiquetas en una imagen

1. Si aún no lo ha hecho:
 - a. Crear o actualizar un usuario de IAM con `AmazonRekognitionFullAccess` y `AmazonS3ReadOnlyAccess` permisos. Para obtener más información, consulte [Paso 1: Configurar una cuenta de AWS y crear un usuario de IAM \(p. 13\)](#).
 - b. Instale y configure la AWS CLI y los AWS SDK. Para obtener más información, consulte [Paso 2: Configurar la AWS CLI y AWS SDK de \(p. 14\)](#).
2. Cargue una imagen que contenga uno o varios objetos (árboles, casas, barcos) en el bucket de S3. La imagen debe estar en formato .jpg o .png.

Para obtener 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 la operación `DetectLabels`.

Java

Este ejemplo muestra una lista de las etiquetas que se han detectado en la imagen de entrada. Sustituir los valores de `bucket` y `photo` con los nombres del bucket de Amazon S3 e 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. Sustituir los valores de `bucket` y `photo` con los nombres del bucket de Amazon S3 e imagen que utilizó en el Paso 2.

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

```
--image '{"S3Object":{"Bucket":"'bucket", "Name":"'file"{}"}'}
```

Java V2

Este código se toma desde el AWS Documentación Ejemplos SDK repositorio de GitHub. Ver el ejemplo completo [aquí](#).

```
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);
    }
}
```

Python

Este ejemplo muestra las etiquetas que se han detectado en la imagen de entrada. Sustituir los valores de `bucket` y `photo` con los nombres del bucket de Amazon S3 e 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.)

import boto3

def detect_labels(photo, bucket):

    client=boto3.client('rekognition')

    response = client.detect_labels(Image={'S3Object':
{'Bucket':bucket,'Name':photo}},
        MaxLabels=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 ("-----")
    print ()
return len(response['Labels'])

def main():
    photo=''
    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 detectadas 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 debucketal nombre del bucket de S3 que contiene el archivo de imagen proporcionado. Cambie el valor deREGIONAL nombre de la región asociada a tu cuenta.

```
// Import required AWS SDK clients and commands for Node.js
import { DetectLabelsCommand } 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"
// Create SNS service object.
const rekogClient = new RekognitionClient({ region: REGION });

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

// Set params
const params = {
  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. Sustituir los valores `debucketyphoto` con los nombres del bucket de Amazon S3 e 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 = "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. Sustituir los valores debucketyphoto con los nombres del bucket de Amazon S3 e imagen que utilizó en el Paso 2.

```
#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.)  
  
# 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
```

Respuesta de ejemplo

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

Cuando realice el `DetectLabels` en una imagen, Amazon Rekognition devuelve información similar a la siguiente respuesta de ejemplo.

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 del `DetectLabels` operación, consulte [Detección de etiquetas](#) (p. 135).

```
{
  "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 cargada 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 API de Amazon Rekognition Image API mediante un archivo cargado a partir de un sistema de archivos local. Puede transferir bytes de imágenes a una operación API de Amazon Rekognition utilizando el [Image \(p. 786\)](#) parámetro de entrada. Dentro de `Image`, especifique la propiedad `Bytes` para transferir bytes de imágenes con codificación en base64.

Bytes de imágenes transferidos a una operación API de Amazon Rekognition utilizando el `Bytes` parámetro de entrada debe tener codificación base64. Los AWS SDK de estos ejemplos utilizan automáticamente imágenes codificadas 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 imágenes \(p. 29\)](#).

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 emplean varios AWS SDK y la AWS CLI para llamar a `DetectLabels`. Para obtener más información sobre la respuesta de la operación `DetectLabels`, consulte [Respuesta DetectLabels \(p. 143\)](#).

Para ver un ejemplo de JavaScript del lado del cliente, consulte [Uso de JavaScript \(p. 45\)](#).

Para detectar las etiquetas en una imagen local

1. Si aún no lo ha hecho:
 - a. Crear o actualizar un usuario de IAM con `AmazonRekognitionFullAccess` y `AmazonS3ReadOnlyAccess` permisos. Para obtener más información, consulte [Paso 1: Configurar una cuenta de AWS y crear un usuario de IAM \(p. 13\)](#).
 - b. Instale y configure la AWS CLI y los AWS SDK. Para obtener más información, consulte [Paso 2: Configurar la AWS CLI y AWSSDK de \(p. 14\)](#).
2. Utilice los siguientes ejemplos para llamar a la operación `DetectLabels`.

Java

El siguiente ejemplo de Java muestra cómo cargar 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 cargar 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 cargar 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 llama a la operación API [DetectFaces](#). 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:  
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 toma desde el AWS Documentación Ejemplos SDK repositorio de GitHub. Ver el ejemplo completo [aquí](#).

```
public static void detectImageLabels(RekognitionClient rekClient, String  
sourceImage) {  
  
    try {  
  
        InputStream sourceStream = new URL("https://images.unsplash.com/  
photo-1557456170-0cf4f4d0d362?ixid=MnwxMjA3fDB8MHxzZWFyY2h8MXx8bGFrZXlxlnwwfHwwfHw  
%3D&ixlib=rb-1.2.1&w=1000&q=80").openStream();
```

```
// InputStream sourceStream = new FileInputStream(sourceImage);
SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);

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

Uso de JavaScript

En el siguiente ejemplo de página web JavaScript permite a un usuario elegir una imagen y ver las edades estimadas de los rostros detectados en la imagen. Las edades estimadas se devuelven mediante una llamada a [the section called “DetectFaces” \(p. 578\)](#).

La imagen elegida se carga con la función `FileReader.readAsDataURL` de JavaScript, 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 decodificar antes de transferirlos a una operación de Amazon Rekognition Image. En este ejemplo se muestra cómo decodificar los bytes de imagen cargados. Si los bytes de imagen codificados no le resultan útiles, use `FileReader.readAsArrayBuffer` en su lugar porque la imagen cargada 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, consulte [Uso de `readAsArrayBuffer` \(p. 48\)](#).

Para ejecutar el ejemplo de JavaScript

1. Cargue el código fuente de ejemplo en un editor.
2. Obtenga el identificador del grupo de identidades de Amazon Cognito. Para obtener más información, consulte [Obtención del identificador del grupo de identidades de Amazon Cognito \(p. 48\)](#).
3. En la función `AnonLog` del código de ejemplo, cambie `IdentityPoolIdToUse` y `RegionToUse` por los valores que anotó en el paso 9 de [Obtención del identificador del grupo de identidades de Amazon Cognito \(p. 48\)](#).
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.

- Elija el iconoExaminar...y elija una imagen que contenga uno o más rostros. 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 de [aws-cognito-sdk.min.js](#) y [amazon-cognito-identity.min.js](#) y, a continuación, guarde el texto de cada uno en un archivo .js diferente.

Código de ejemplo JavaScript

En el siguiente ejemplo de código se utiliza JavaScript V2. Para ver un ejemplo en JavaScript V3, consulte [el ejemplo de la AWS Documentación Ejemplos SDK repositorio de GitHub](#).

```
<!--
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;
        }
    });
}

function ProcessImage() {
    var file = document.getElementById("fileToUpload").files[0];
    var reader = new FileReader();
    reader.onload = function (e) {
        DetectFaces(e.target.result);
    }
    reader.readAsArrayBuffer(file);
}
```

```
        }
    });
}
//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("data:image/jpeg;base64,")[1]);
            } catch (e) {
                jpg = false;
            }
            if (jpg == false) {
                try {
                    image = atob(e.target.result.split("data:image/png;base64,")[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(imageBytes);
        };
    })(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 readAsArrayBuffer

El siguiente fragmento de código es una implementación alternativa de `ProcessImage` en el código de ejemplo, utilizando JavaScript V2. Utiliza `readAsArrayBuffer` para cargar una imagen y llama a `DetectFaces`. Porque `readAsArrayBuffer` no codifica en base64 el archivo cargado, no es necesario decodificar 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);  
}
```

Obtención del identificador del 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 imagen de Amazon Rekognition. 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 [Grupo 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 Create new identity pool.
3. En Identity pool name* (Nombre de grupo de identidades), escriba un nombre para su grupo de identidades.
4. En Unauthenticated identities (Identidades sin autenticar), elija Enable access to unauthenticated identities (Habilitar el acceso a identidades sin autenticar).
5. Elija Create Pool.
6. Elija View Details (Ver detalles) y anote el nombre de rol de las identidades sin autenticar.
7. Elija Allow.
8. En Platform (Plataforma), elija JavaScript.
9. En Get AWS Credentials (Obtener credenciales de AWS), anote los valores de `AWS.config.region` y `IdentityPoolId` que se muestran en el fragmento de código.
10. Abra la consola de IAM en <https://console.aws.amazon.com/iam/>.
11. Seleccione Roles (Roles) en el panel de navegación.
12. Elija el nombre de rol que anotó en el paso 6.
13. En la pestaña Permissions (Permisos), elija Attach policies (Asociar políticas).
14. Elija AmazonRekognitionReadOnlyAccess.
15. Elija Attach Policy (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 [the section called “DetectFaces” \(p. 578\)](#) devuelve un cuadro delimitador ([the section called “BoundingBox” \(p. 740\)](#)) 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`: ancho del cuadro delimitador expresada proporcionalmente respecto a la anchura total de la imagen.

Cada propiedad de `BoundingBox` tiene un valor comprendido 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 su ubicación y tamaño correctos, deberá multiplicar los valores de `BoundingBox` por la anchura o altura de la imagen (según el valor que desee) para obtener los valores en 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 [Obtener orientación de imagen y coordenadas de cuadro delimitador \(p. 57\)](#).

En los siguientes ejemplos se indica cómo mostrar un cuadro delimitador que enmarca los rostros detectados llamando a [DetectFaces \(p. 578\)](#). En los ejemplos se da por hecho que las imágenes tienen una orientación de 0 grados. Los ejemplos también muestran cómo descargar la imagen desde un bucket de Amazon S3.

Para mostrar un cuadro delimitador

1. Si aún no lo ha hecho:
 - a. Crear o actualizar un usuario de IAM con `AmazonRekognitionFullAccess` y `AmazonS3ReadOnlyAccess` permisos. Para obtener más información, consulte [Paso 1: Configurar una cuenta de AWS y crear un usuario de IAM \(p. 13\)](#).
 - b. Instale y configure la AWS CLI y los AWS SDK. Para obtener más información, consulte [Paso 2: Configurar la AWS CLI y AWS SDK de \(p. 14\)](#).
2. Utilice los siguientes ejemplos para llamar a la operación `DetectFaces`.

Java

Cambie el valor de `bucket` en 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` en 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).

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

def show_faces(photo,bucket):

    client=boto3.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"
    photo="photo"

    faces_count=show_faces(photo,bucket)
    print("faces detected: " + str(faces_count))

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

Java V2

Este código se toma desde el AWS Documentación Ejemplos SDK repositorio de GitHub. Ver el ejemplo completo [aquí](#).

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

```
public static void displayAllFaces(S3Client s3,
                                    RekognitionClient rekClient,
                                    String sourceImage,
                                    String bucketName) {
    int height = 0;
    int width = 0;

    byte[] data = getObjectTypeBytes(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);
        DisplayFacesFrame panel = new DisplayFacesFrame(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();
        } 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);
        byte[] data = objectBytes.asByteArray();
        return data;
    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return null;
}

public DisplayFacesFrame(BufferedImage bufImage) throws Exception {
    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 = 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 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));
    }
}
```

Obtener orientación de imagen y coordenadas de cuadro delimitador

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 mostrar una imagen correctamente en su aplicación, tiene que conocer la orientación de la imagen. Es posible que 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 dichas coordenadas. Las operaciones de detección de rostros de Amazon Rekognition con imágenes devuelven las coordenadas del cuadro delimitador para cada rostro detectado, pero no estiman las coordenadas de los archivos.jpg sin metadatos Exif.

En los siguientes ejemplos se indica 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.

Cómo encontrar 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 \(p. 740\)](#).

Para mostrar un cuadro delimitador alrededor de un rostro, similar al cuadro mostrado en la imagen siguiente, en su aplicación, utilice las coordenadas del 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 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: Obtener orientación de imagen y coordenadas de cuadro delimitador de una imagen

En los siguientes ejemplos se indica cómo se utiliza AWS SDK para obtener los datos de orientación de imagen Exif y las coordenadas del cuadro delimitador para los famosos detectados por `elRecognizeCelebrities`.

Note

Support la estimación de la orientación de la imagen mediante el `orientationCorrection` campo ha cesado en agosto de 2021. Los valores devueltos para este campo incluidos en una respuesta de API siempre serán NULL.

Java

En este ejemplo se carga una imagen desde el sistema de archivos local y se llama `aRecognizeCelebrities`, determina la altura y la anchura de la imagen y calcula las coordenadas del cuadro delimitador de la cara 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 utiliza la Biblioteca de imágenes PIL/Pillow para obtener la anchura y altura 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 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 toma desde el AWS Documentación Ejemplos SDK repositorio de GitHub. Ver el ejemplo completo [aquí](#).

```
public static void recognizeAllCelebrities(RekognitionClient rekClient, String
sourceImage) {

    try {
        BufferedImage image = null;
        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 = 0;
    float top = 0;

    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();
            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:
            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()));
}
```

Trabajar con 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, por ejemplo, medios de comunicación/entretenimiento y seguridad pública. Anteriormente, la exploración de vídeos para detectar objetos o personas habría llevado 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, información general de la API y ejemplos de uso de Amazon Rekognition Video.

Temas

- [Tipos de análisis \(p. 64\)](#)
- [Descripción general de la API de Amazon Rekognition Video \(p. 64\)](#)
- [Llamar a las operaciones de Amazon Rekognition Video \(p. 66\)](#)
- [Configuración de Amazon Rekognition Video \(p. 70\)](#)
- [Análisis de un vídeo almacenado en un bucket de Amazon S3 con Java o Python \(SDK\) \(p. 72\)](#)
- [Análisis de un vídeo con la AWS Command Line Interface \(p. 89\)](#)
- [Referencia: Notificación de los resultados del análisis \(p. 91\)](#)
- [Solución de problemas de Amazon Rekognition Video \(p. 92\)](#)

Tipos de análisis

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

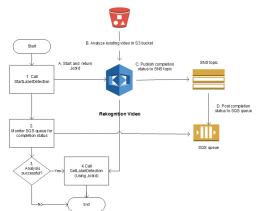
- [Segmentos de vídeo \(p. 338\)](#)
- [Etiquetas \(p. 135\)](#)
- [Contenido para adultos insinuante y explícito \(p. 298\)](#)
- [Text \(p. 315\)](#)
- [Famosos \(p. 275\)](#)
- [Rostros \(p. 149\)](#)
- [Personas \(p. 243\)](#)

Para obtener más información, consulte [Cómo funciona Amazon Rekognition \(p. 4\)](#).

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 \(p. 701\)](#). El estado de realización de la solicitud se publica un tema de Amazon Simple Notification Service (Amazon SNS). Para obtener el estado de realización del tema de Amazon SNS, puede utilizar una cola de Amazon Simple Queue Service (Amazon SQS) o una función Lambda. Una vez que disponga del estado de realización, llame a una operación `Get`, como [GetLabelDetection \(p. 626\)](#), para obtener los resultados de la solicitud.

El siguiente diagrama 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. También puede utilizar una función AWS Lambda.



El proceso es el mismo para otras operaciones de Amazon Rekognition Video. En la siguiente tabla se enumeran los `Start`y `Get`para cada una de las operaciones de Amazon Rekognition sin almacenamiento.

Detección	Operación START	Operación GET
Segmentos de vídeo	StartSegmentDetection (p. 712)	GetSegmentDetection (p. 635)
Etiquetas	StartLabelDetection (p. 701)	GetLabelDetection (p. 626)
Contenido para adultos explícito o insinuante	StartContentModeration (p. 689)	GetContentModeration (p. 611)
Texto	StartTextDetection (p. 718)	GetTextDetection (p. 640)
Famosos	StartCelebrityRecognition (p. 685)	GetCelebrityRecognition (p. 605)
Rostros	StartFaceDetection (p. 693)	GetFaceDetection (p. 615)
Personas	StartPersonTracking (p. 705)	GetPersonTracking (p. 630)

Para las operaciones 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 acerca del uso de Amazon Rekognition Video, consulte [Llamar a las operaciones de Amazon Rekognition Video \(p. 66\)](#). 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 \(SDK\) \(p. 72\)](#). Para ver ejemplos de AWS CLI, consulte [Análisis de un vídeo con la AWS Command Line Interface \(p. 89\)](#).

Formatos de vídeo y almacenamiento

Las operaciones de Amazon Rekognition pueden analizar vídeos que están almacenados en buckets de Amazon S3. 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 codificar el vídeo sea H.264.

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 \(p. 644\)](#). A continuación, puede utilizar [StartFaceSearch \(p. 697\)](#) para comenzar a buscar asíncronamente rostros en la colección. Puede utilizar [GetFaceSearch \(p. 620\)](#) para obtener los resultados de la búsqueda. Para obtener más información, consulte [Búsqueda de rostros en vídeos almacenados \(p. 234\)](#). La búsqueda de personas es un ejemplo de operación de Amazon Rekognition basada en almacenamiento. Para obtener más información, consulte [Operaciones de API basadas en almacenamiento de información \(p. 9\)](#).

También puede buscar personas en un vídeo en streaming. Para obtener más información, consulte [Uso de vídeos en streaming \(p. 94\)](#).

Llamar a las operaciones de Amazon Rekognition Video

Amazon Rekognition Video es una API asíncrona que puede utilizar para analizar los vídeos que se almacenan en un bucket de Amazon Simple Storage Service (Amazon S3). El análisis de un vídeo comienza llamando a Amazon Rekognition VideoStartoperación, tales como[StartPersonTracking \(p. 705\)](#). 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 de Amazon Simple Queue Service (Amazon SQS) o unAWS Lambda para obtener el estado de la realización de la solicitud de análisis de vídeo desde el tema de Amazon SNS. Por último, obtiene los resultados de la solicitud de análisis de vídeo llamando a Amazon RekognitionGetoperación, tales como[GetPersonTracking \(p. 630\)](#).

La información de las secciones siguientes utiliza las operaciones de detección de etiquetas para mostrar cómo Amazon Rekognition Video detecta 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,[StartFaceDetection \(p. 693\)](#) y [StartPersonTracking \(p. 705\)](#). El ejemplo[Análisis de un vídeo almacenado en un bucket de Amazon S3 con Java o Python \(SDK\) \(p. 72\)](#) muestra cómo analizar un vídeo mediante la utilización de una cola de Amazon SQS para obtener el estado de realización del tema de Amazon SNS. También se utiliza como base para otros ejemplos de Amazon Rekognition Video, como por ejemplo[Recorridos de las personas \(p. 243\)](#). Para ver ejemplos de AWS CLI, consulte [Análisis de un vídeo con la AWS Command Line Interface \(p. 89\)](#).

Temas

- [Comenzar el análisis de vídeo \(p. 66\)](#)
- [Obtener el estado de finalización de una solicitud de análisis de Amazon Rekognition Video \(p. 67\)](#)
- [Obtención de los resultados del análisis de Amazon Rekognition Video \(p. 68\)](#)

Comenzar el análisis de vídeo

Puede iniciar una solicitud de detección de etiquetas de Amazon Rekognition Video llamando a [StartLabelDetection \(p. 701\)](#). El siguiente es un ejemplo de una solicitud JSON que ha transferido StartLabelDetection.

```
{  
    "Video": {  
        "S3Object": {  
            "Bucket": "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 entradaVideo proporciona el nombre de 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 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 enlace 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 de Amazon Rekognition a los temas de Amazon SNS creando un rol de servicio de IAM. Para obtener más información, consulte [Configuración de Amazon Rekognition Video \(p. 70\)](#).

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 diferente `Start` ya que obtendrá resultados impredecibles de Amazon Rekognition.

La respuesta a la operación `StartLabelDetection` es un identificador de trabajo (`JobId`).

Usar `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 empiezas demasiados trabajos simultáneamente, llama `StartLabelDetection` para `RaiseLimitExceeded` (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 encuentras `esolimitExceeded` Las excepciones se producen con picos de actividad, considere la posibilidad de utilizar una cola de Amazon SQS para administrar las solicitudes entrantes. Póngase en contacto con el servicio de soporte de AWS si descubre que el número medio de solicitudes simultáneas no se puede administrar con una cola de Amazon SQS y sigue recibiendo excepciones `LimitExceeded`.

Obtener el estado de finalización 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": "270c1cc5e1d0ea2fbc59d97cb69a72a5495da75851976b14a1nnnnnnnnnnnn",  
    "Status": "SUCCEEDED",  
    "API": "StartLabelDetection",  
    "JobTag": "DetectingLabels",  
    "Timestamp": 1510865364756,  
    "Video": {  
        "S3ObjectName": "video.mp4",  
        "S3Bucket": "bucket"  
    }  
}
```

```
}
```

Para obtener más información, consulte [Referencia: Notificación de los resultados del análisis \(p. 91\)](#).

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 un AWS Lambda que escribe 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, sonde 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 \(SDK\) \(p. 72\)](#). 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 al Amazon Rekognition VideoGet. Esto se debe a que Amazon Rekognition Video limita elGetOperación si se realizan demasiadas solicitudes. Si está procesando varios vídeos simultáneamente, es más sencillo y más eficaz supervisar 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 conserva 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
}
}
```

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`.

Configuración de Amazon Rekognition Video

Para utilizar la API de Amazon Rekognition Video con vídeos almacenados, debe configurar el usuario de IAM y un rol de servicio de IAM para que puedan acceder 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 \(SDK\) \(p. 72\)](#), no es necesario que realice los pasos 3, 4, 5 y 6. El ejemplo incluye código para 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 \(p. 71\)](#) para el paso 3.

Para configurar Amazon Rekognition Video

1. Configurar unAWSuenta para acceder a Amazon Rekognition Video. Para obtener más información, consulte [Paso 1: Configurar una cuenta de AWS y crear un usuario de IAM \(p. 12\)](#).

Asegúrese de que el usuario tiene al menos los permisos siguientes:

- `AmazonSQSFullAccess`
 - `AmazonRekognitionFullAccess`
 - `AmazonS3FullAccess`
 - `AmazonSNSFullAccess`
2. Instale y configure el SDK de AWS necesario. Para obtener más información, consulte [Paso 2: Configurar laAWS CLlAyAWSSDK de \(p. 14\)](#).
 3. Cree un tema de Amazon SNS mediante el uso de la[Consola de Amazon SNS](#). Anexe `AmazonRekognition` al nombre del tema. 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 enlace de AWS que está utilizando.

4. Creación de una cola estándar de Amazon SQS mediante el uso de la[Consola de Amazon SQS](#). Anote el ARN de la cola.
5. Suscriba la cola al tema que creó en el paso 3.
6. Conceder permiso al tema de Amazon SNS y enviar mensajes a la cola de Amazon SQS.
7. Cree un rol de servicio de IAM para dar a Amazon Rekognition Video acceso a los temas de Amazon SNS. Anote el nombre de recurso de Amazon (ARN) del rol de servicio. Para obtener más información, consulte [Acceso a varios temas de Amazon SNS \(p. 71\)](#).
8. Añada la siguiente política insertada al usuario de IAM que ha creado en el paso 1:

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Sid": "MySid",  
            "Effect": "Allow",  
            "Action": "iam:PassRole",  
            "Resource": "arn:Service role ARN from step 7"  
        }  
    ]  
}
```

Asigne a la política insertada un nombre de su elección.

9. Ahora puede ejecutar los ejemplos de [Análisis de un vídeo almacenado en un bucket de Amazon S3 con Java o Python \(SDK\) \(p. 72\)](#) y [Análisis de un vídeo con la AWS Command Line Interface \(p. 89\)](#).

Acceso a varios temas de Amazon SNS

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

Puede otorgar acceso a Amazon Rekognition Video a varios temas de Amazon SNS mediante el AmazonRekognitionServiceRole política de permisos y anteponer los nombres de los temas con AmazonRekognition—por ejemplo, AmazonRekognitionMyTopicName.

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

1. [Creación de 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 aparecer la política de permisos AmazonRekognitionServiceRole. AmazonRekognitionServiceRole proporciona a Amazon Rekognition Video acceso a temas de Amazon SNS que tienen 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 siguiente política de. Reemplazartopicarncon el Nombre de recurso de Amazon (ARN) del tema de Amazon SNS deseado.

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": [  
                "sns:Publish"  
            ],  
            "Resource": "topicarn"  
        }  
    ]  
}
```

2. Creación de 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 [Llamar a las operaciones de Amazon Rekognition Video \(p. 66\)](#). No está limitado a la utilización de una cola de Amazon SQS. Por ejemplo, puede utilizar una función de AWS Lambda para obtener el estado de realización. 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 \(p. 701\)](#).
6. Obtenga el estado de realización en 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 erróneos de Amazon SQS para examinarlos posteriormente.
7. Obtenga y muestre los resultados del análisis de vídeo llamando a [GetLabelDetection \(p. 626\)](#).

Requisitos previos

El código de ejemplo de este procedimiento se proporciona en Java y Python. Debe tener instalado el SDK de AWS adecuado. Para obtener más información, consulte [Introducción a Amazon Rekognition \(p. 12\)](#). 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. Configurar el acceso de los usuarios a Amazon Rekognition Video y configurar el acceso de Amazon Rekognition Video a Amazon SNS. Para obtener más información, consulte [Configuración de Amazon Rekognition Video \(p. 70\)](#). 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 un bucket de Amazon S3. Para realizar pruebas, cargue un vídeo con una duración inferior a 30 segundos.

Para obtener 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:

- Reemplazar `roleArn` con el ARN del rol de servicio de IAM que creó en el paso 7 de [Para configurar Amazon Rekognition Video \(p. 70\)](#).
- Reemplace los valores de `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();
sqc = 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(snsTopicArn)))
);

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: " + sqsSubscriptionArn);
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`:

- Reemplazar `roleArn` con el ARN del rol de servicio de IAM que creó en el paso 7 de [Para configurar Amazon Rekognition Video \(p. 70\)](#).
- Reemplace los valores de `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.)  
  
import boto3  
import json  
import sys  
import time  
  
  
class VideoDetect:  
    jobId = ''  
    rek = boto3.client('rekognition')  
    sqs = boto3.client('sqs')  
    sns = boto3.client('sns')  
  
    roleArn = ''  
    bucket = ''  
    video = ''  
    startJobId = ''  
  
    sqsQueueUrl = ''  
    snsTopicArn = ''  
    processType = ''  
  
    def __init__(self, role, bucket, video):  
        self.roleArn = role  
        self.bucket = bucket  
        self.video = video  
  
    def GetSQSMessageSuccess(self):  
  
        jobFound = False  
        succeeded = False  
  
        dotLine=0  
        while jobFound == False:  
            sqsResponse = self.sqs.receive_message(QueueUrl=self.sqsQueueUrl,  
MessageAttributeName=['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})

    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')

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

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

        sqsQueueArn = attrs['QueueArn']

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

        #Authorize SNS to write SQS queue
        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 = ''  
bucket = ''  
video = ''  
  
analyzer=VideoDetect(roleArn, bucket,video)  
analyzer.CreateTopicandQueue()  
  
analyzer.StartLabelDetection()  
if analyzer.GetSQSMessageSuccess()==True:  
    analyzer.GetLabelDetectionResults()  
  
analyzer.DeleteTopicandQueue()  
  
if __name__ == "__main__":  
    main()
```

Node.Js

En el siguiente código de muestra:

- Sustituir el valor de `REGION` con el nombre de la región operativa de tu cuenta.
- Sustituir el valor de `bucket` con el nombre del bucket de Amazon S3 que contiene el archivo de vídeo.
- Sustituir el valor de `videoName` con el nombre del archivo de vídeo de su bucket de Amazon S3.
- Reemplazar `roleArn` con el ARN del rol de servicio de IAM que creó en el paso 7 de [Para configurar Amazon Rekognition Video \(p. 70\)](#).

```
// snippet-start:[sqS.JavaScript.queues.createQueueV3]  
// 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 }  
from "@aws-sdk/client-rekognition";  
import { stdout } from "process";  
  
// Set the AWS Region.  
const REGION = "region-name"; //e.g. "us-east-1"  
// Create SNS service object.  
const sqsClient = new SQSClient({ region: REGION });  
const snsClient = new SNSClient({ region: REGION });  
const rekClient = new RekognitionClient({ region: REGION });  
  
// 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 attrsResponse = await sqsClient.send(new
GetQueueAttributesCommand({QueueUrl: sqsQueueUrl, AttributeNames: ['QueueArn']}))
    const attrs = attrsResponse.Attributes
    console.log(attrs)
    const queueArn = attrs.QueueArn
    // subscribe SQS queue to SNS topic
    const subscribed = await snsClient.send(new SubscribeCommand({TopicArn:
topicArn, Protocol:'sq', Endpoint: queueArn}))
    const policy = {
      Version: "2012-10-17",
      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()
            // Search 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 "SUCCEEDED", 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 toma de la AWS Ejemplos de la documentación del SDK repositorio de GitHub. Ver el ejemplo completo [aquí](#).

```
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=null;
    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();
} catch (Exception 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 videoMetaDataAdapter=labelDetectionResult.videoMetadata();

            System.out.println("Format: " + videoMetaDataAdapter.format());
            System.out.println("Codec: " + videoMetaDataAdapter.codec());
            System.out.println("Duration: " + videoMetaDataAdapter.durationMillis());
            System.out.println("FrameRate: " + videoMetaDataAdapter.frameRate());

            List<LabelDetection> detectedLabels= labelDetectionResult.labels();
            for (LabelDetection detectedLabel: detectedLabels) {
                long seconds=detectedLabel.timestamp();
                Label label=detectedLabel.label();
                System.out.println("Millisecond: " + Long.toString(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);
}
}
```

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 \(p. 146\)](#).

Análisis de un vídeo con la AWS Command Line Interface

Puede utilizar el AWS Command Line Interface(AWS CLI) para llamar a las operaciones de Amazon Rekognition Video. El patrón de diseño es el mismo que con la API de Amazon Rekognition Video con AWS SDK for Java o otros SDK de AWS. Para obtener más información, consulte [Descripción general de la API de Amazon Rekognition Video \(p. 64\)](#). Los siguientes procedimientos muestran cómo utilizar la 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 `--notification-channel` parámetro 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. Luego sondeos [recibir mensaje](#) para obtener el estado de realización a partir de la cola de Amazon SQS.

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 los resultados del análisis \(p. 91\)](#). 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`.

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 \(SDK\) \(p. 72\)](#).

Requisitos previos

Para ejecutar este procedimiento, necesita tener AWS CLI instalado. Para obtener más información, consulte [Introducción a Amazon Rekognition \(p. 12\)](#). 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 cargar un vídeo

1. Configurar el acceso de los usuarios a Amazon Rekognition Video y configurar el acceso de Amazon Rekognition Video a Amazon SNS. Para obtener más información, consulte [Configuración de Amazon Rekognition Video \(p. 70\)](#).
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 obtener 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 comando de AWS CLI para comenzar la detección de etiquetas en un vídeo.

```
awsrekognition start-label-detection --video  
  "S3Object={Bucket=bucketname,Name=videofile}" \  
  --notification-channel "SNSTopicArn=TopicARN,RoleArn=RoleARN" \  
  --endpoint-url Endpoint \  
  --region us-east-1
```

Actualice los siguientes valores:

- Cambiobucketnameyvideofileal nombre de bucket de Amazon S3 y de archivo que especificó en el paso 2.
 - Cambie us-east-1 por la región de AWS que está utilizando.
 - CambioTopicARNal ARN del tema de Amazon SNS que ha creado en el paso 3 de [Configuración de Amazon Rekognition Video \(p. 70\)](#).
 - CambioRoleARNal ARN del rol de servicio de IAM que creó en el paso 7 de [Configuración de Amazon Rekognition Video \(p. 70\)](#).
 - Si es necesario, puede especificar laendpoint-url. La CLI de AWS debe determinar automáticamente la URL del endpoint adecuada en función de la región proporcionada. Sin embargo, si está utilizando un endpointdesde su VPC privada, es posible que tenga que especificar laendpoint-url. LaEndpoints de servicio de AWSresource enumera la sintaxis para especificar las URL de endpoint y los nombres y códigos de cada región.
2. Anote el valor de JobId en la respuesta. La respuesta tiene un aspecto similar a la del siguiente ejemplo JSON.

```
{  
  "JobId": "547089ce5b9a8a0e7831afa655f42e5d7b5c838553f1a584bf350ennnnnnnnnnn"
```

3. Escriba código para sondear la cola de Amazon SQS para ver el archivo JSON de estado de realización (mediante [recibir mensaje](#)).
4. Escriba código para extraer el campo Status del JSON de estado de realización.
5. Si el valor de Status es SUCCEEDED, ejecute el siguiente comando de AWS CLI para mostrar los resultados de detección de etiqueta.

```
aws rekognition get-label-detection --job-id JobID \  
--region us-east-1
```

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 enlace 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 los resultados del análisis

Amazon Rekognition publica los resultados de una solicitud de análisis de Amazon Rekognition Video, incluido el estado de realización, en un tema de Amazon Simple Notification Service (Amazon SNS). Para obtener la notificación de un tema de Amazon SNS, utilice una cola de Amazon Simple Queue Service o unAWS Lambda función. Para obtener más información, consulte [the section called “Llamar a las operaciones de Amazon Rekognition Video” \(p. 66\)](#). Para ver un ejemplo, consulte [Análisis de un vídeo almacenado en un bucket de Amazon S3 con Java o Python \(SDK\) \(p. 72\)](#).

La carga está en el siguiente formato JSON:

```
{  
    "JobId": "String",  
    "Status": "String",  
    "API": "String",  
    "JobTag": "String",  
}
```

```
    "Timestamp": Number,  
    "Video": {  
        "S3ObjectName": "String",  
        "S3Bucket": "String"  
    }  
}
```

Name (Nombre)	Descripción
JobId	El identificador del trabajo. Coincide con un identificador de trabajo que se devuelve desde una operación Start, como por ejemplo StartPersonTracking (p. 705) .
Estado	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.
JobTag	Identificador del trabajo. Especifique JobTag en una llamada a una operación Start, como por ejemplo StartLabelDetection (p. 701) .
Marca temporal	La marca de hora Unix cuando el trabajo terminó.
Vídeo	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": "videobucket"  
    }  
}
```

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 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 [Suscribirse a un tema](#).

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 \(p. 70\)](#).
- Confirme que el rol de servicio de IAM que está utilizando puede publicar en el tema de Amazon SNS mediante las credenciales del rol. Utilice los siguientes pasos:
 - 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 mediante la [consola de administración de AWS](#). Por ejemplo:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "rekognition.amazonaws.com",
                "AWS": "arn:User ARN"
            },
            "Action": "sts:AssumeRole",
            "Condition": {}
        }
    ]
}
```

- Asuma el rol: `aws sts assume-role --role-arn arn:Role ARN --role-session-name SessionName --profile RekognitionUser`
- Publique en el tema de Amazon SNS:`aws sns publish --topic-arn arn:Topic ARN --message "Hello World!" --region us-east-1 --profile RekognitionUser`

Si el comando de la CLI de AWS funciona, recibirá el mensaje (en su bandeja de entrada, si se ha suscrito al tema por correo electrónico). Si no recibe el mensaje:

- Compruebe que ha configurado Amazon Rekognition Video. Para obtener más información, consulte [Configuración de Amazon Rekognition Video \(p. 70\)](#).
- 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 otorgar 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 \(p. 71\)](#).
 - Si utiliza un rol de servicio de IAM para otorgar a Amazon Rekognition Video acceso a varios temas de SNS, compruebe que está utilizando el tema correcto y que el nombre de este viene precedido por `AmazonRekognition`. Para obtener más información, consulte [Acceso a varios temas de Amazon SNS \(p. 71\)](#).
 - Si utiliza un AWS Lambda, confirme que la función de Lambda está suscrita al tema de Amazon SNS correcto. Para obtener más información, consulte [Invocación de funciones Lambda mediante notificaciones de Amazon SNS](#).

- Si suscribe una cola de Amazon SQS a su 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 [Conceder permiso al tema de Amazon SNS y enviar mensajes a la cola de Amazon SQS](#).

Necesito ayuda adicional para solucionar problemas de tema de Amazon SNS

Puede usar AWS X-Ray con Amazon SNS para rastrear y analizar los mensajes que pasan por su aplicación. Para obtener más información, consulte [Amazon SNS y AWS X-Ray](#).

Para obtener ayuda adicional, puedes publicar tu pregunta en el [Foro de Amazon Rekognition](#) o considere inscribirse en [AWS Soporte técnico de](#).

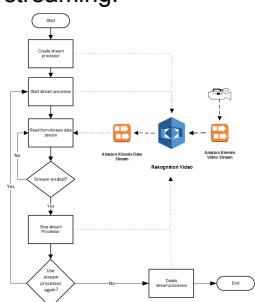
Uso de vídeos en streaming

Puede usar Amazon Rekognition Video para detectar y reconocer rostros en vídeo en streaming. Un caso de uso típico es cuando se desea detectar un rostro conocido en una transmisión de vídeo. 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 una transmisión de datos de Kinesis y, a continuación, los lee la aplicación cliente. Amazon Rekognition Video proporciona un procesador de secuencias ([CreateStreamProcessor \(p. 537\)](#)) que puede utilizar para comenzar y administrar el análisis de vídeo en streaming.

Note

La API de streaming Amazon Rekognition Video está disponible solo en las siguientes regiones: EE.UU. Este (Norte de Virginia), EE.UU. Oeste (Oregón), Asia Pacífico (Tokio), UE (Fráncfort) y UE (Irlanda).

El siguiente diagrama muestra cómo Amazon Rekognition Video detecta y reconoce rostros en vídeo en streaming.



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 Video Streams](#).
- Un procesador de streaming Amazon Rekognition Video para administrar el análisis del vídeo en streaming. Para obtener más información, consulte [Analizar la transmisión de vídeos con los procesadores de transmisión de Amazon Rekognition Video \(p. 97\)](#).
- Un consumidor de flujos de datos de Kinesis para leer los resultados del análisis que Amazon Rekognition Video envía al flujo de datos de Kinesis. Para obtener más información, consulte [Consumidores de Kinesis Data Streams](#).

Esta sección contiene información sobre cómo escribir una aplicación que crea la transmisión de vídeo de Kinesis y la transmisión de datos de Kinesis, transmite vídeo a Amazon Rekognition Video y consume los resultados de los análisis. Si está transmitiendo desde un archivo codificado Matroska (MKV), puede utilizar el [PutMedia](#) para transmitir el vídeo de origen en la secuencia de vídeo de Kinesis que creó. Para obtener más información, consulte [PutMedia API Example](#). De lo contrario, puede utilizar Gstreamer, un software de marco multimedia de terceros, e instalar un complemento de Amazon Kinesis Video Streams que transmite vídeo desde la cámara de un dispositivo.

Temas

- [Configuración de los recursos de Amazon Rekognition Video y Amazon Kinesis \(p. 95\)](#)
- [Transmisión mediante un complemento de GStreamer \(p. 107\)](#)
- [Leer resultados de análisis de vídeo en streaming \(p. 109\)](#)
- [Referencia: Registro de reconocimiento facial de Kinesis \(p. 113\)](#)
- [Solución de problemas de streaming de vídeo \(p. 117\)](#)

Configuración de los recursos de Amazon Rekognition Video y Amazon Kinesis

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 \(p. 181\)](#). El siguiente procedimiento describe los pasos que debe realizar para aprovisionar la transmisión de vídeo de Kinesis y la transmisión de datos de Kinesis que se utilizarán para reconocer rostros en vídeo en streaming.

Requisitos previos

Para ejecutar este procedimiento, necesita tener AWS SDK for Java instalado. Para obtener más información, consulte [Introducción a Amazon Rekognition \(p. 12\)](#). 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](#) en la [IAM User Guide](#).

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 a su Kinesis Video Streams sus secuencias de datos de Kinesis. Anote el ARN. Para obtener más información, consulte [Acceso a sus transmisiones de vídeo de Kinesis y secuencias de datos de Kinesis \(p. 96\)](#).
2. Cree una colección ([p. 185](#)) y anote el identificador de la colección que haya utilizado.
3. Indexe los rostros ([p. 204](#)) que desee buscar en la colección que ha creado en el paso 2.
4. Creación de una secuencia de vídeo de Kinesis y anote el Nombre de recurso de Amazon (ARN) de la secuencia.
5. [Creación de una secuencia de datos de Kinesis](#). Anexe al nombre de la transmisión AmazonRekognition y anote el ARN de la transmisión.
6. [Cree el procesador de streaming \(p. 99\)](#). Pase lo siguiente como parámetros a the section called “CreateStreamProcessor” ([p. 537](#)): un nombre de su elección, el ARN de streaming de vídeo de Kinesis (paso 4), el ARN de streaming de datos de Kinesis (paso 5) y el identificador de la colección (paso 2).
7. [Comience el procesador de streaming \(p. 99\)](#) utilizando el nombre del procesador de streaming que eligió en el paso 6.

Note

Debe iniciar el procesador de streaming solo después de haber verificado que puede ingerir contenido multimedia en la transmisión de vídeo de Kinesis.

Consulte la siguiente página:

- Si está transmitiendo desde una fuente codificada Matroska (MKV), utilice el[PutMedia](#)para transmitir el vídeo de origen en la secuencia de vídeo de Kinesis que creó. Para obtener más información, consulte [PutMedia API Example](#).
- Si estás transmitiendo desde la cámara de un dispositivo, consulta[Transmisión mediante un complemento de GStreamer \(p. 107\)](#).

Dar acceso a Amazon Rekognition Video a sus transmisiones de Kinesis

Usas unAWS Identity and Access Management(IAM) para proporcionar acceso de lectura de Amazon Rekognition Video a las secuencias de vídeo de Kinesis y acceso de escritura a los flujos de datos de Kinesis.

Acceso a sus transmisiones de vídeo de Kinesis y secuencias de datos de Kinesis

IAM proporciona elRekognitioncaso de uso del rol de servicio que, cuando se utiliza con elAmazonRekognitionServiceRolepolítica de permisos, puede escribir en varios flujos de datos de Kinesis y leer desde todas sus secuencias de vídeo de Kinesis. Para dar acceso de escritura de Amazon Rekognition Video a varios flujos de datos de Kinesis, puede añadir los nombres de los flujos de datos de Kinesis conAmazonRekognition—por ejemplo,AmazonRekognitionMyDataStreamName.

Para dar acceso a Amazon Rekognition Video a la transmisión de vídeo de Kinesis y a la transmisión de datos de Kinesis

1. [Creación de 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 el iconoAmazonRekognitionServiceRolepolítica de permisos, que otorga a acceso de escritura de Amazon Rekognition Video a streaming de datos de Kinesis que tienen el prefijo deAmazonRekognitiony acceso de lectura a todas sus transmisiones de vídeo de Kinesis.
2. Anote el nombre de recurso de Amazon (ARN) del rol de servicio. Lo necesita para comenzar las operaciones de análisis de vídeo.

Acceso a transmisiones individuales de Kinesis

Puede crear una política de permisos que permita a Amazon Rekognition Video acceder a secuencias de vídeo individuales de Kinesis y flujos de datos de Kinesis.

Para dar acceso a Amazon Rekognition Video a una secuencia de vídeo individual de Kinesis y a una secuencia de datos de Kinesis

1. [Cree una nueva política de permisos con el editor de políticas de JSON de IAM](#)y utilice la siguiente política. Reemplazardata-arncon el ARN del flujo de datos de Kinesis deseado yvideo-arncon el ARN de la transmisión de vídeo de Kinesis deseada.

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": [  
                "kinesis:PutRecord",  
                "kinesis:PutRecords"  
            ],  
            "Resource": "data-arn"  
        },  
        {  
            "Effect": "Allow",  
            "Action": [  
                "kinesisvideo:GetDataEndpoint",  
                "kinesisvideo:GetMedia"  
            ],  
            "Resource": "video-arn"  
        }  
    ]  
}
```

2. [Creación de 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.

Analizar la transmisión de vídeos con los procesadores de transmisión de Amazon Rekognition Video

El análisis de un streaming de vídeo comienza iniciando un procesador de streaming Amazon Rekognition Video y transmitiendo vídeo a Amazon Rekognition Video. Un procesador de streaming Amazon Rekognition Video le permite iniciar, detener y administrar procesadores de streaming. Crea un procesador de streaming llamando a [CreateStreamProcessor \(p. 537\)](#). Los parámetros de solicitud incluyen los nombres de recursos de Amazon (ARN) para la transmisión de vídeo de Kinesis, la transmisión de datos de Kinesis y el identificador de la colección que se utiliza para reconocer rostros en el vídeo en streaming. También incluye el nombre que especifica para el procesador de streaming.

Comienza a procesar un vídeo llamando a la operación

Comienza el procesamiento de un procesador de streaming. Crea un procesador de streaming llamando a [CreateStreamProcessor \(p. 537\)](#). Para contar `StartStreamProcessor` qué procesador de flujo iniciar, utilice el valor de la `Name` campo especificado en la llamada a `CreateStreamProcessor`.

Sintaxis de la solicitud

```
{  
    "Name": "string"  
}
```

Parámetros de solicitud

La solicitud acepta los siguientes datos en formato JSON.

Name (p. 716)

El nombre del procesador de secuencias que se va a iniciar el procesamiento.

Type: Cadena

Restricciones de longitud: Longitud mínima de 1. La longitud máxima es de 128 caracteres.

Patrón: [a-zA-Z0-9_.\-\-]+

Obligatorio: Sí

Elementos de respuesta

Si la acción se realiza correctamente, el servicio devuelve una respuesta HTTP 200 con un cuerpo HTTP vacío.

Errores

AccessDeniedException

No tiene autorización para realizar la acción.

Código de estado HTTP: 400

InternalServerError

Amazon Lex ha tenido un problema de servicio. Pruebe la llamada de nuevo.

Código de estado HTTP: 500

InvalidArgumentException

El parámetro de entrada infringió una restricción. Valide los parámetros antes de llamar a la operación de la API de nuevo.

Código de estado HTTP: 400

ProvisionedThroughputExceededException

El número de solicitudes ha superado su límite de rendimiento. Si necesita aumentar este límite, póngase en contacto con Amazon Rekognition.

Código de estado HTTP: 400

ResourceInUseException

El recurso especificado ya se está utilizando.

Código de estado HTTP: 400

ResourceNotFoundException

El recurso especificado en la solicitud no se encuentra.

Código de estado HTTP: 400

ThrottlingException

Amazon Lex no puede procesar temporalmente la solicitud. Pruebe la llamada de nuevo.

Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- [AWS Command Line Interface](#)
- [SDK de AWS para .NET](#)
- [AWS SDK para C++](#)
- [AWS SDK para Go](#)
- [AWSSDK para Java V2](#)
- [AWS SDK para JavaScript](#)
- [SDK de AWS para PHP V3](#)
- [SDK de AWS para Python](#)
- [SDK de AWS para Ruby V3](#)

(p. 716). Para obtener la información de estado de un procesador de streaming, llame a [DescribeStreamProcessor](#) (p. 569). Otras operaciones que puede llamar son [StopStreamProcessor](#) (p. 724) para detener un procesador de streaming y [DeleteStreamProcessor](#) (p. 554) para eliminar un procesador de streaming. Para obtener una lista de los procesadores de streaming en su cuenta, llame a [ListStreamProcessors](#) (p. 666).

Después de que el procesador de streaming comienza a ejecutarse, transmita el vídeo en Amazon Rekognition Video a través del streaming de vídeo de Kinesis que especificó en `enCreateStreamProcessor`. Uso del SDK de Kinesis Video Streams [PutMedia](#) para enviar vídeo a la transmisión de vídeo de Kinesis. Para ver un ejemplo, consulte [PutMedia API Example](#).

Para obtener más información sobre cómo la aplicación puede consumir los resultados de análisis de Amazon Rekognition Video, consulte [Leer resultados de análisis de vídeo en streaming](#) (p. 109).

Creación del procesador de secuencias de Amazon Rekognition Video

Antes de poder analizar un vídeo en streaming, cree un procesador de streaming Amazon Rekognition Video ([CreateStreamProcessor](#) (p. 537)). El procesador de streaming contiene información sobre el streaming de datos de Kinesis y el streaming 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"  
}
```

Etiquetado del procesador de secuencias de Amazon Rekognition Video

Puede identificar, organizar, buscar y filtrar los procesadores de streaming de Amazon Rekognition mediante etiquetas. Cada etiqueta es una marca que consta de una clave y un valor definidos por el usuario.

Temas

- [Agregar etiquetas a un nuevo procesador de secuencias \(p. 100\)](#)
- [Incorporación de etiquetas a un procesador de streaming existente \(p. 100\)](#)
- [Lista de etiquetas en un procesador de secuencias \(p. 101\)](#)
- [Eliminar etiquetas de un procesador de secuencias \(p. 101\)](#)

Agregar etiquetas a un nuevo procesador de secuencias

También puede añadir etiquetas a un procesador de streaming a medida que lo cree utilizando la `CreateStreamProcessor`. Especifique una o varias etiquetas en la `Tags` parámetro de entrada de matriz. A continuación, se muestra un ejemplo de JSON para la `CreateStreamProcessor` solicitud 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::nnnnnnnnnnnn:role/roleWithKinesisPermission",  
    "Settings": {  
        "FaceSearch": {  
            "CollectionId": "collection-with-100-faces",  
            "FaceMatchThreshold": 85.5  
        },  
        "Tags": {  
            "Dept": "Engineering",  
            "Name": "Ana Silva Carolina",  
            "Role": "Developer"  
        }  
    }  
}
```

Incorporación de etiquetas a un procesador de streaming existente

Para añadir una o más etiquetas a un procesador de streaming existente, utilice la `TagResource`. Especifique el nombre de recurso de Amazon (ARN) del procesador de streaming (`ResourceArn`) y las etiquetas (`Tags`) que desea agregar. En el ejemplo siguiente, se muestra cómo añadir dos etiquetas.

```
aws rekognition tag-resource --resource-arn resource-arn \
--tags '{"key1":"value1","key2":"value2"}'
```

Note

Si no conoce el nombre de recurso de Amazon del procesador de secuencias, puede utilizar `elDescribeStreamProcessor`.

[Lista de etiquetas en un procesador de secuencias](#)

Para enumerar las etiquetas asociadas con un procesador de streaming de, utilice `laListTagsForResource`y especificar el ARN del procesador de secuencias (`ResourceArn`). La respuesta es un mapa de las claves y los valores de las etiquetas que se asocian al procesador de streaming especificado.

```
aws rekognition list-tags-for-resource --resource-arn resource-arn
```

La salida muestra una lista de etiquetas adjuntas al procesador de secuencias:

```
{
  "Tags": {
    "Dept": "Engineering",
    "Name": "Ana Silva Carolina",
    "Role": "Developer"
  }
}
```

[Eliminar etiquetas de un procesador de secuencias](#)

Para eliminar una o varias etiquetas de un procesador de streaming de, utilice `laUntagResource`. Especifique el ARN del modelo (`ResourceArn`) y las teclas de etiqueta (`Tag-Keys`) que desea eliminar.

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

Si lo desea, puede especificar claves de etiqueta en este formato:

```
--tag-keys key1,key2
```

[Inicio del procesador de secuencias de Amazon Rekognition Video](#)

Se comienza a analizar el vídeo en streaming llamando a [StartStreamProcessor \(p. 716\)](#) 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 transmisión (ejemplo de Java V2)

El siguiente código de ejemplo muestra cómo llamar a diversas operaciones de procesador, como [CreateStreamProcessor \(p. 537\)](#) y [StartStreamProcessor \(p. 716\)](#), utilizando el AWS SDK for Java versión 2.

Este código se toma de la [AWS Documentación de ejemplos del SDK de repositorio de GitHub](#). Ver el ejemplo completo [aquí](#).

```
public static void listStreamProcessors(RekognitionClient rekClient) {

    ListStreamProcessorsRequest request = ListStreamProcessorsRequest.builder()
        .maxResults(15)
        .build();

    ListStreamProcessorsResponse listStreamProcessorsResult =
rekClient.listStreamProcessors(request);

    //List all stream processors (and state) returned from Rekognition.
    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);

    // Display the results.
    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 transmisión (ejemplo de Java V1)

El siguiente código de ejemplo muestra cómo llamar a diversas operaciones de procesador, como [CreateStreamProcessor \(p. 537\)](#) y [StartStreamProcessor \(p. 716\)](#), utilizando Java V1. El ejemplo incluye una clase de administrador de procesador de streaming (StreamManager) que proporciona métodos para llamar a operaciones de procesador de streaming. La clase de inicio (Starter) crea un objeto StreamManager y llama a diversas 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());
        }
    }
}
```

Clase StreamManager

```
//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(streamProcessorOutput)
    .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());
    }
}
}
```

Transmisión de vídeo en Amazon Rekognition Video

Para transmitir vídeo en Amazon Rekognition Video, utilice el SDK de Amazon Kinesis Video Streams para crear y utilizar una transmisión de vídeo de Kinesis. La `PutMedia` operación escribe datos de videofragments en una transmisión de vídeo de Kinesis que consume Amazon Rekognition Video. Cada fragmento de datos de vídeo suele tener una longitud de 2 a 10 segundos y contiene una secuencia autónoma de fotogramas de vídeo. 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 la transmisión de vídeo de Kinesis, Kinesis Video Streams asigna un número único al fragmento. Para ver un ejemplo, consulte [PutMedia API Example](#).

Transmisión mediante un complemento de GStreamer

Amazon Rekognition Video puede analizar una transmisión de vídeo en directo desde la cámara de un dispositivo. Para acceder a la entrada multimedia desde una fuente de dispositivo, necesitas instalar GStreamer. GStreamer es un software de marco multimedia de terceros que conecta fuentes de medios y herramientas de procesamiento en canalizaciones de flujo de trabajo. También es necesario instalar el [Plugin de productor de Amazon Kinesis Video Streams](#) para Gstreamer. En este proceso se supone 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 \(p. 95\)](#).

Paso 1: Instale Gstreamer

Descargue e instale Gstreamer, un software de plataforma multimedia de terceros. Puede utilizar un software de administración de paquetes como Homebrew ([Gstreamer en Homebrew](#)) o obtenerlo directamente desde el [Sitio web de escritorio gratuito](#).

Verifique la instalación correcta de Gstreamer iniciando un feed 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 productores de Amazon Kinesis Video Streamse](#) instala el complemento Kinesis Video Streams Gstreamer.

Cree un directorio y clone el código fuente desde el repositorio de Github. Asegúrese de incluir la--recursiveparámetro.

```
$ git clone --recursive https://github.com/awslabs/amazon-kinesis-video-streams-producer-sdk-cpp.git
```

Siga la[instrucciones proporcionadas por la biblioteca](#)para configurar y compilar el proyecto.

Asegúrese de utilizar los comandos específicos de la plataforma para su sistema operativo. Usar-DBUILD_GSTREAMER_PLUGIN=ONparámetro cuando ejecutacmakepara 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 debido a que falta un paquete, compruebe que el paquete está instalado y en su PATH. Si aparece un error de «no se puede ejecutar el programa compilado C» durante la compilación, vuelva a ejecutar el comando build. A veces, no se encuentra el compilador C correcto.

Para verificar la instalación del complemento de Kinesis Video Streams, ejecute este 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, es posible que deba convertir la fuente multimedia en un códec aceptable para Kinesis Video Streams. Para determinar las especificaciones y las capacidades de formato de los dispositivos conectados actualmente a su equipo, ejecute este comando.

```
$ gst-device-monitor-1.0
```

Para comenzar a transmitir, inicie Gstreamer con el siguiente comando de ejemplo y agregue sus credenciales e información de Amazon Kinesis Video Streams. Debe utilizar las claves de acceso y la región para el rol de servicio de IAM que creó mientras [otorgar acceso a Amazon Rekognition a sus transmisiones de Kinesis](#). Para obtener más información sobre claves de acceso, consulte [Administración de las claves de acceso de los usuarios de IAM](#) en la [IAM User Guide](#). Además, puede ajustar los parámetros de argumentos de formato de vídeo según lo requiera su uso y disponibles desde 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 lanzamiento, consulte [Comandos de lanzamiento de GStreamer de ejemplo](#).

Note

Si el comando de lanzamiento finaliza con un error de no negociación, compruebe la salida del Monitor de dispositivos y asegúrese de que el `videoconvert` los valores de parámetros son capacidades válidas de su dispositivo.

Verás una fuente de vídeo de la cámara de tu dispositivo en la transmisión de vídeo de Kinesis después de unos segundos. Para empezar a detectar y hacer coincidir rostros con Amazon Rekognition, inicie el procesador de secuencias de Amazon Rekognition Video. Para obtener más información, consulte [Analizar la transmisión de vídeos con los procesadores de transmisión de Amazon Rekognition Video \(p. 97\)](#).

Leer resultados de 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 [Lectura de datos de un flujo de datos de Kinesis](#). 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 cada fotograma que se le transfiere a través del streaming de vídeo de Kinesis.

Un registro de fotograma que se envía a una transmisión de datos de Kinesis contiene información acerca del fragmento de streaming de vídeo 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 [Referencia: Registro de reconocimiento facial de Kinesis \(p. 113\)](#).

La biblioteca de analizadores de Amazon Kinesis Video Streams contiene pruebas de ejemplo que consumen los resultados de Amazon Rekognition Video y los integra con el flujo de vídeo original de Kinesis. Para obtener más información, consulte [Visualización de resultados Rekognition con Kinesis Video Streams localmente \(p. 112\)](#).

Amazon Rekognition Video transmite información de análisis de Amazon Rekognition Video al flujo 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": 0.40833333  
            }  
        }  
    ]  
}]
```

```
        "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 acerca del streaming de vídeo de Kinesis que se utiliza para transmitir vídeo en Amazon Rekognition Video. Para obtener más información, consulte [InputInformation \(p. 115\)](#).
- **StreamProcessorInformation**— Información de estado para el procesador de streaming de Amazon Rekognition Video. El único valor posible para el campo **Status** es **RUNNING**. Para obtener más información, consulte [StreamProcessorInformation \(p. 116\)](#).
- **FaceSearchResponse**: contiene información sobre rostros en el vídeo en streaming que coincide con rostros en la colección de entrada.[FaceSearchResponse \(p. 116\)](#) contiene un[DetectedFace \(p. 116\)](#) objeto, que es un rostro que se detectó en el fotograma de vídeo analizado. Para cada rostro detectado, la matriz **MatchedFaces** contiene una matriz de objetos de rostro coincidentes ([MatchedFace \(p. 117\)](#)) 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 streaming de vídeo de Kinesis a los fotogramas analizados que se envían a la transmisión 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 a la transmisión 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 origen de Kinesis.

La técnica que utiliza para asignar la transmisión de vídeo de Kinesis a la transmisión 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).

Mapeo cuando estás transmitiendo contenido multimedia en directo

Para asignar un fotograma de flujo de vídeo de Kinesis a un marco de flujo de datos de Kinesis

1. Establezca el parámetro de entrada **FragmentTimeCodeType** de la operación [PutMedia](#) en **RELATIVE**.
2. Llame a [PutMedia](#) para enviar medios en directo a la transmisión de vídeo de Kinesis.
3. Cuando reciba un registro del reconocimiento facial de Kinesis de la transmisión de datos de Kinesis, almacene los valores **deProducerTimestamp**, **frameOffsetInSeconds** desde las [KinesisVideo \(p. 115\)](#).
4. Calcule la marca temporal correspondiente al fotograma de streaming de vídeo de Kinesis añadiendo **laProducerTimestamp**, **frameOffsetInSeconds** valores de campo juntos.

Mapeo cuando estás transmitiendo contenido multimedia archivado

Para asignar un fotograma de flujo de vídeo de Kinesis a un marco de flujo de datos de Kinesis

1. Llamada[PutMedia](#)para enviar medios archivados 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 la transmisión de datos de Kinesis, almacene elFrameOffsetInSecondsvalor de campo de la[KinesisVideo \(p. 115\)](#).
4. Calcule la asignación mediante elFrameOffsetInSecondsyFragmentNumbervalores almacenados en los pasos 2 y 3.FrameOffsetInSecondses la diferencia del fragmento con elFragmentNumberque se envía 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 de resultados Rekognition con Kinesis Video Streams localmente

Puede ver los resultados de Amazon Rekognition Video mostrados en su feed desde Amazon Kinesis Video Streams mediante las pruebas de ejemplo de la biblioteca de analizadores de Amazon Kinesis Video Streams que se proporcionan en[KinesisVideo - Ejemplos Rekognition](#). La[KinesisVideoRekognitionIntegrationExample](#) muestra cuadros delimitadores sobre las caras detectadas y procesa el vídeo localmente a través de JFrame. En este proceso se supone que ha conectado correctamente una entrada multimedia de una cámara de dispositivo a una transmisión de vídeo de Kinesis e inició un procesador Amazon Rekognition Stream. Para obtener más información, consulte[Transmisión mediante un complemento de GStreamer \(p. 107\)](#).

Paso 1: Instalación de la biblioteca de analizadores de Kinesis Video Streams

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: Configuración de la prueba de integración de Kinesis Video Streams y Rekognition

Abra el archivo `KinesisVideoRekognitionIntegrationExampleTest.java`. Retirar `@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 \(p. 95\)](#). Si va a transmitir vídeo a la transmisión de vídeo de Kinesis, elimine el `inputStream` parámetro.

Consulte el siguiente código de ejemplo:

```
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.getInputStream("bezos_vogels.mkv"))
.build();
```

Paso 3: Ejecución de la prueba de integración de Kinesis Video Streams y Rekognition

Asegúrese de que la transmisión de vídeo de Kinesis reciba entrada multimedia si está transmitiendo en ella y comience a analizar su transmisión con un procesador Amazon Rekognition Video Stream en ejecución. Para obtener más información, consulte [Analizar la transmisión de vídeos con los procesadores de transmisión de Amazon Rekognition Video \(p. 97\)](#). Ejecute la `KinesisVideoRekognitionIntegrationExampleTest` clase como prueba JUnit. Tras un breve retraso, se abre una nueva ventana con una fuente 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 el Id. de imagen externa (el nombre del archivo) especificado en este formato para que las etiquetas de los cuadros delimitadores muestren un texto significativo: Nombre de persona 1 de confianza, Nombre de persona 2 Intruder, Nombre de persona 3 neutral, etc. Las etiquetas también se pueden codificar por colores y se pueden personalizar en el archivo `FaceType.java`.

Referencia: Registro de reconocimiento facial de Kinesis

Amazon Rekognition Video puede reconocer rostros en un vídeo en streaming. Para cada fotograma analizado, Amazon Rekognition Video devuelve un registro de fotograma de JSON a una transmisión de datos de Kinesis. Amazon Rekognition Video no analiza cada fotograma que se le transfiere a través del streaming 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 secuencia de datos de Kinesis. Para obtener más información, consulte [Uso de vídeos en streaming \(p. 94\)](#).

Note

La API Amazon Rekognition Video funciona comparando las caras de la transmisión 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 acerca del streaming de vídeo de Kinesis que se utiliza para transmitir vídeo en Amazon Rekognition Video.

Tipo: objeto [InputInformation \(p. 115\)](#)

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 \(p. 116\)](#)

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 \(p. 116\)](#)

InputInformation

Información acerca de una transmisión de vídeo de origen que utiliza Amazon Rekognition Video. Para obtener más información, consulte [Uso de vídeos en streaming \(p. 94\)](#).

KinesisVideo

Información acerca del streaming de vídeo de Kinesis que transmite el vídeo de origen en Amazon Rekognition Video. Para obtener más información, consulte [Uso de vídeos en streaming \(p. 94\)](#).

StreamArn

El nombre de recurso de Amazon (ARN) del flujo de vídeo de Kinesis.

Type: Cadena

FragmentNumber

El fragmento del vídeo en streaming que contiene el fotograma que representa este registro.

Type: Cadena

ProducerTimestamp

La marca temporal Unix del lado del productor del fragmento. Para obtener más información, consulte [PutMedia](#).

Type: Número

ServerTimestamp

La marca temporal Unix del lado del servidor del fragmento. Para obtener más información, consulte [PutMedia](#).

Type: Número

FrameOffsetInSeconds

El desfase del fotograma (en segundos) dentro del fragmento.

Type: Número

StreamProcessorInformation

Información de estado acerca del procesador de streaming.

Estado

El estado actual del procesador de streaming. El único valor posible es RUNNING.

Type: 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 \(p. 537\)](#). Para obtener más información, consulte [Uso de vídeos en streaming \(p. 94\)](#).

DetectedFace

Detalles de un rostro detectado en un fotograma de vídeo analizado.

Tipo: objeto [DetectedFace \(p. 116\)](#)

MatchedFaces

Una matriz de detalles de rostros en una colección que coincide con el rostro detectado en `DetectedFace`.

Tipo: matriz de objetos [MatchedFace \(p. 117\)](#)

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 \(p. 117\)](#).

BoundingBox

Las coordenadas del cuadro delimitador de un rostro que se ha detectado dentro de un fotograma de vídeo analizado. El objeto `BoundingBox` tiene las mismas propiedades que el objeto `BoundingBox` que se ha utilizado para el análisis de la imagen.

Tipo: objeto [BoundingBox](#) (p. 740)

Confianza

El nivel de confianza (de 1 a 100) que tiene Amazon Rekognition Video de que el rostro detectado es realmente un rostro. 1 es la confianza más baja, 100 es la mayor.

Type: Número

Referencias

Una matriz de referencias faciales.

Tipo: matriz de objetos [Landmark](#) (p. 795)

Postura

Indica la postura del rostro tal como determina su cabeceo, balanceo y desviación.

Tipo: objeto [Pose](#) (p. 806)

Calidad

Identifica el brillo y la nitidez de la imagen del rostro.

Tipo: objeto [ImageQuality](#) (p. 788)

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](#) (p. 116).

Tipo: objeto [Face](#) (p. 771)

Similitud

El nivel de confianza (de 1 a 100) de la coincidencia de rostros. 1 es la confianza más baja, 100 es la mayor.

Type: Número

Solución de problemas de streaming de vídeo

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 \(p. 118\)](#)
- [No sé si he configurado correctamente mi procesador de streaming \(p. 118\)](#)
- [Mi procesador de streaming no está devolviendo resultados \(p. 119\)](#)
- [El estado de mi procesador de streaming es FAILED \(p. 120\)](#)

- [Mi procesador de streaming no está devolviendo los resultados esperados \(p. 122\)](#)

No sé si mi procesador de streaming se ha creado correctamente

Utilice el siguiente comando de AWS CLI para obtener una lista de los procesadores de streaming y su estado actual.

```
aws rekognition list-stream-processors
```

Puede obtener detalles adicionales utilizando el siguiente comando de AWS CLI. 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 el 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 \(p. 119\)](#). Si el estado es `FAILED`, consulte [El estado de mi procesador de streaming es FAILED \(p. 120\)](#).

```
aws rekognition describe-stream-processor --name stream-processor-name
```

2. Si el procesador de streaming está en ejecución, ejecute el siguiente comando Bash o PowerShell para leer los datos de la secuencia 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 [Paso 3: Obtenga el récord](#).
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 [Dar acceso a Amazon Rekognition Video a sus transmisiones de Kinesis \(p. 96\)](#).

También puede utilizar el «`kinesis-process-record`» AWS Lambda blueprint para registrar mensajes desde el flujo de datos de Kinesis en CloudWatch para una visualización continua. Esto incurre en costos adicionales para AWS Lambda y 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 \(p. 118\)](#).

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 streaming con el siguiente comando de AWS CLI.

```
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 \(p. 120\)](#).
 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 [Uso de vídeos en streaming \(p. 94\)](#).
  - c. Si sigue teniendo problemas, póngase en contacto con AWS Support.
5. Si el valor de `Status` es `RUNNING`, consulte [Motivo 3: No hay datos activos en la secuencia de vídeo de Kinesis \(p. 119\)](#).
- ### Motivo 3: No hay datos activos en la secuencia de vídeo de Kinesis
- Para comprobar si hay datos activos en la transmisión de vídeo de Kinesis
1. Inicie sesión en la AWS Management Console y abra la consola de Amazon Kinesis Video Streams en <https://console.aws.amazon.com/kinesisvideo/>.
  2. Seleccione la transmisión de vídeo de Kinesis que es la entrada para el procesador de streaming Amazon Rekognition.
  3. Si la vista previa indica que no hay datos en la transmisión, a continuación, no hay ningún dato en la secuencia de entrada para que Amazon Rekognition Video lo procese.
- Para obtener información acerca de la producción de vídeo con Kinesis Video Streams, consulte [Bibliotecas de productores de Kinesis Video Streams](#).
- 119

## El estado de mi procesador de streaming es FAILED

Puede comprobar el estado de un procesador de streaming mediante el siguiente comando de AWS CLI.

```
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 AWS Management Console y abra la consola de IAM 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 tiene la política de permisos AmazonRekognitionServiceRole.
4. Si el rol no existe o no tiene los permiso adecuados, consulte [Dar acceso a Amazon Rekognition Video a sus transmisiones de Kinesis \(p. 96\)](#).
5. Inicie el procesador de streaming con el siguiente comando de AWS CLI.

```
aws rekognition start-stream-processor --name stream-processor-name
```

### Error: «Acceso denegado a Kinesis VideooAcceso denegado a Kinesis Data»

La función no tiene acceso a las operaciones de API de Kinesis Video StreamsGetMedia y GetDataEndpoint. Es posible que tampoco tenga acceso a las operaciones de API de Kinesis Data StreamsPutRecord y PutRecords.

Solución de problemas de permisos de API

1. Inicie sesión en la AWS Management Console y abra la consola de IAM en <https://console.aws.amazon.com/iam/>.
2. Abra el rol y asegúrese de que tiene la siguiente política de permisos asociada.

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Allow",
 "Action": [
 "kinesis:PutRecord",
 "kinesis:PutRecords"
],
 "Resource": "data-arn"
 },
 {
 "Effect": "Allow",
 "Action": [
 "kinesisvideo:GetDataEndpoint",
 "kinesisvideo:GetMedia"
],
 "Resource": "media-arn"
 }
]
}
```

```
 "Resource": "video-arn"
 }
}
}
```

3. Si falta alguno de los permisos, actualice la política. Para obtener más información, consulte [Dar acceso a Amazon Rekognition Video a sus transmisiones de Kinesis \(p. 96\)](#).

### Error: «Stream *input-video-stream-name* no existe»

La entrada de streaming de vídeo de Kinesis en el procesador de streaming no existe o no está configurada correctamente.

Para solucionar problemas de la transmisión 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 transmisión de vídeo de Kinesis está en la misma Región que el procesador de streaming.

Si el procesador de streaming no está configurado correctamente, elimínelo con el siguiente comando de AWS CLI.

```
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 secuencias de Amazon Rekognition Video \(p. 99\)](#).

### Error: «Colección no encontrada»

La colección Amazon Rekognition que utiliza el procesador de streaming para comparar rostros no existe o se está utilizando una colección equivocada.

Para confirmar la colección

1. Utilice el siguiente comando de AWS CLI para determinar si la colección necesaria existe. Cambie `region` a la región de AWS 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 \(p. 181\)](#).

2. En la llamada a [the section called “CreateStreamProcessor” \(p. 537\)](#), compruebe que el valor del parámetro de entrada de `CollectionId` es correcto.
3. Inicie el procesador de streaming con el siguiente comando de AWS CLI.

```
aws rekognition start-stream-processor --name stream-processor-name
```

## Error: «Stream **output-kinesis-data-stream-name** en cuenta **account-id** no se encuentra»

La secuencia de datos de Kinesis de salida que utiliza el procesador de streaming no existe en su cuenta de AWS o no se encuentra en la misma Región de AWS que el procesador de streaming.

Para solucionar problemas de la secuencia de datos de Kinesis

1. Utilice el siguiente comando de AWS CLI para determinar si existe la secuencia de datos de Kinesis. Cambie `region` a la Región de AWS en la que está utilizando el procesador de streaming.

```
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 secuencia de datos de Kinesis no existe, podría existir en otra Región de AWS. 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: Creación de una secuencia de datos](#).
  - b. Inicie el procesador de streaming con el siguiente comando de AWS CLI.

```
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 \(p. 181\)](#)
- [Recomendaciones para la configuración de la cámara \(streaming de vídeo\) \(p. 134\)](#)

# Control 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 del error \(p. 122\)](#)
- [Mensajes y códigos de error \(p. 123\)](#)
- [Administración de errores en la aplicación \(p. 127\)](#)

## Componentes del error

Cuando el programa envía una solicitud, Amazon Rekognition intenta procesarla. Si la solicitud se realiza 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.`).

Los SDK de AWS se encargan de transmitir los errores a la aplicación, para que pueda adoptar las medidas apropiadas. 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-7e48-11e9-9ad1-854a4567eb71","Message":"Unable 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 devueltas por 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

: Message An error failed (AccessDeniedException) when calling the <Operation>operation: Usuario: <User ARN>no está autorizado a realizar: <Operation>en el recurso: <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

: Message No se pudo programar el trabajo GroupFaces. 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

: Message El ClientRequestToken: <Token> que ha solicitado ya está en uso.

Se ha reutilizado un parámetro de entrada ClientRequestToken con una operación, pero al menos uno de los demás parámetros de entrada es distinto de la llamada anterior a la operación.

¿Reintentar? No

#### ImageTooLargeException

: Message El tamaño de la imagen es demasiado grande.

La imagen de entrada tamaño supera el límite permitido. Si estás llamando [the section called "DetectProtectiveEquipment" \(p. 592\)](#), el tamaño o la resolución de la imagen superan el límite permitido. Para obtener más información, consulte [Directrices y cuotas de Amazon Rekognition \(p. 847\)](#).

¿Reintentar? No

#### InvalidImageFormatException

: Message 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 de Amazon Rekognition \(p. 847\)](#).

¿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

: Message 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

#### InvalidS3ObjectException

##### Mensajes:

- La solicitud tiene un objeto de S3 no válido.
- No se pueden obtener los metadatos de objeto desde S3. Compruebe la clave de objeto, la región o los permisos de acceso.

Amazon Rekognition no puede obtener acceso al objeto de S3 que se especificó en la solicitud. Para obtener más información, consulte [Configurar el acceso a S3: AWS S3 Administración del acceso](#). 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>.
- <Number of Open Jobs>open Trabajos para el usuario <User ARN>Límite máximo: <Maximum Limit>

Se ha superado un límite de servicio de Amazon Rekognition. Por ejemplo, si inicia demasiados trabajos de Amazon Rekognition Video al mismo tiempo, las llamadas para iniciar operaciones, como `StartLabelDetection`, levante una `LimitExceededException` excepción (código de estado HTTP: 400) hasta que el número de trabajos ejecutados simultáneamente se encuentre 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 del servicio de Amazon Rekognition](#).

Para solicitar un aumento del límite, siga las instrucciones de la sección llamada “[Crear un caso para cambiar las cuotas TPS](#)” (p. 849).

¿Reintentar? Sí

#### ResourceAlreadyExistsException

: Message El identificador de la colección: <Collection Id> 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

: Message Varios mensajes en función de la llamada a la API.

El recurso especificado no existe.

¿Reintentar? No

#### ThrottlingException

: Message Desaceleración; aumento repentino de la tasa de solicitudes.

Su tasa de aumento de solicitudes es demasiado rápida. Reduzca la tasa de solicitudes y aumentela gradualmente. Le recomendamos que interrumpa exponencialmente y vuelva a intentarlo. De forma predeterminada, los SDK de AWS usan una lógica de reintentos automático y retardo exponencial.

Para obtener más información, consulte [Reintentos de error y retardo exponencial en AWS](#) y [Retardo exponencial y fluctuación](#).

¿Reintentar? Sí

#### VideoTooLargeException

: Message Tamaño del vídeo en bytes: <Video Size>es mayor que el límite máximo de: <Max Size>bytes.

El tamaño del archivo o la duración del medio suministrado es demasiado grande. Para obtener más información, consulte [Directrices y cuotas de Amazon Rekognition \(p. 847\)](#).

¿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)

: Message Error interno de servicio

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, los SDK de AWS usan una lógica de reintentos automático y retardo exponencial. Para obtener más información, consulte [Reintentos de error y retardo exponencial en AWS](#) y [Retardo exponencial y fluctuación](#).

¿Reintentar? Sí

#### ThrottlingException (HTTP 500)

: Message Service Unavailable

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, los SDK de AWS usan una lógica de reintentos automático y retardo exponencial. Para obtener más información, consulte [Reintentos de error y retardo exponencial en AWS](#) y [Retardo exponencial y fluctuación](#).

¿Reintentar? Sí

## Administración 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`.

Los AWS SDK llevan a cabo sus propios reintentos y comprobaciones de errores. Si se produce algún error al utilizar uno de los SDK de AWS, su código y descripción pueden ayudarle a solucionar el problema.

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 Amazon Rekognition no ha podido procesarla y ha devuelto una respuesta de error.
- `AmazonClientException`: esta excepción se produce si el cliente no ha podido obtener una respuesta de un servicio o sí la ha obtenido pero no ha podido analizarla.

## Uso de Amazon Rekognition como servicio autorizado de FedRAMP

La AWSEI programa de conformidad con FedRAMP de incluye a Amazon Rekognition como servicio autorizado para 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 enlace 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 enlace de FIPS, consulte [Información general sobre FIPS 140-2](#).

Puede utilizar el AWS Command Line Interface(AWS CLI) o uno de los SDK de AWS para especificar el punto de enlace que utiliza Amazon Rekognition.

Para obtener información sobre los puntos de enlace que se pueden utilizar con Amazon Rekognition, consulte [Regiones y puntos de enlace de Amazon Rekognition](#).

A continuación se muestran ejemplos de la [Listado de colecciones \(p. 192\)](#) tema en el Guía para desarrolladores de Amazon Rekognition. Se modifican para especificar la región y el punto de enlace FIPS a través de los cuales se accede a Amazon Rekognition.

#### Java

Para Java, utilice el `withEndpointConfiguration` cuando construya el cliente de Amazon Rekognition. Este ejemplo muestra las colecciones que tiene que utilizar el punto de enlace 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);
 } while (listCollectionsResult != null && listCollectionsResult.getNextToken() != null);
 }
}
```

```
 List < String > collectionIds = listCollectionsResult.getCollectionIds();
 for (String resultId: collectionIds) {
 System.out.println(resultId);
 }
 } while (listCollectionsResult != null && listCollectionsResult.getNextToken() !=
 =
 null);

}
}
```

#### AWS CLI

Para el registroAWS CLI, utilice el--endpoint-urlpara especificar el punto de enlace a través del cual se obtiene acceso a Amazon Rekognition. Este ejemplo muestra las colecciones que tiene que utilizar el punto de enlace de FIPS en la región de EE. UU. Este (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 enlace que quiera especificar. Este ejemplo muestra las colecciones que tiene que utilizar el punto de enlace de FIPS en la región de EE. UU. Oeste (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, imágenes de entrada y videos

Esta sección contiene información acerca de las prácticas recomendadas para el uso de Amazon Rekognition.

## Temas

- [Latencia de operación de imagen de Amazon Rekognition \(p. 131\)](#)
- [Recomendaciones para imágenes de entrada de comparación facial \(p. 131\)](#)
- [Recomendaciones para la configuración de la cámara \(imagen y vídeo\) \(p. 132\)](#)
- [Recomendaciones para la configuración de la cámara \(almacenado y streaming de vídeo\) \(p. 133\)](#)
- [Recomendaciones para la configuración de la cámara \(streaming de vídeo\) \(p. 134\)](#)

## Latencia de operación de imagen de Amazon Rekognition

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 imagen de Amazon Rekognition.
- 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 cargada 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, hacer referencia a ella en una operación de Amazon Rekognition Image probablemente sea más rápido que transferir bytes de imagen a la operación.

## Recomendaciones para imágenes de entrada de comparación facial

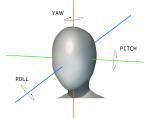
Los modelos que se utilizan en las operaciones de reconocimiento de rostros están diseñados para funcionar con una amplia variedad de posturas, expresiones faciales, rangos de edad, rotaciones, condiciones de iluminación y tamaños. Le recomendamos que aplique las siguientes directrices cuando elija las fotos de referencia de [CompareFaces \(p. 515\)](#) o si desea agregar rostros a una colección utilizando [IndexFaces \(p. 644\)](#).

- 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.
- 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 se indexen al menos cinco imágenes de la persona: directamente, cara girada a la izquierda con una rotación sobre la rotación sobre el eje X de 45 grados o menos, cara inclinada hacia abajo con un cabeceo de 30 grados o menos, y cara inclinada hacia arriba con un cabeceo 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, las cinco imágenes de John Doe se pueden marcar en la colección con identificadores de imagen externos, como `John_Doe_1.jpg` , . . . , `John_Doe_5.jpg`.
- 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 términos de resolución. Amazon Rekognition puede reconocer rostros 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 presenten 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.
- Utilice imágenes que 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](#) (p. 578) para determinar el brillo y la nitidez de un rostro.
- Asegúrese de indexar las imágenes recientes de los rostros.

## Recomendaciones para la configuración de la cámara (imagen y vídeo)

Las siguientes recomendaciones son adicionales a las que se indican en [Recomendaciones para imágenes de entrada de comparación facial](#) (p. 131).



- 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: hay tres mediciones del ángulo de la cámara: el ángulo de rotación sobre el eje X, la rotación sobre el eje Y.
  - Cabeceo: recomendamos que sea inferior a 30 grados cuando la cámara mira hacia abajo y menos de 45 grados cuando la cámara mira hacia arriba.
  - Roll: no hay un requisito mínimo para este parámetro. Amazon Rekognition puede manejar cualquier cantidad de rollo.
  - Guincha — Recomendamos la rotación sobre el eje X de menos de 45 grados en ambos sentidos.

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: la resolución mínima recomendada del rostro de 50 x 50 píxeles debe basarse en el parámetro de la cámara. 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: el ángulo de rotación sobre el eje X de la cámara debe basarse en

## Recomendaciones para la configuración de la cámara (almacenado y streaming de vídeo)

Las siguientes recomendaciones son adicionales a las que se indican en [Recomendaciones para la configuración de la cámara \(imagen y vídeo\) \(p. 132\)](#).

- 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.
- Velocidad de fotogramas frente a la resolución de fotogramas: si la velocidad de bits del codificador está restringida, recomendamos dar opción a la mayor resolución de fotogramas respecto a la velocidad de fotogramas, para obtener mejores resultados de búsqueda de rostros. De este modo, se asegura de que Amazon Rekognition obtenga el fotograma de mejor 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 1,5mbps, 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 para la configuración de la cámara (streaming de vídeo)

La siguiente recomendación es adicional a las que se indican en [Recomendaciones para la configuración de la cámara \(almacenado y streaming de vídeo\) \(p. 133\)](#).

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 en 1,5 Mbps, el ancho de banda disponible de la red debe 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 para adaptarse a los nervios debido a las conexiones caídas entre una cámara y el concentrador. El valor real dependerá de la fiabilidad de la red interna.

# Detección de etiquetas

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, escena, acción o concepto encontrado en una imagen o vídeo basado en su contenido. Por ejemplo, una foto de personas en una playa tropical podría contener etiquetas como «Palm Tree» (objeto), «Beach» (escena), «Running» (acción) y «Outdoors» (concepto).

Para descargar la lista completa de etiquetas y cuadros delimitadores de objetos compatibles con Amazon Rekognition, haga clic en[aquí](#).

## Note

Amazon Rekognition hace predicciones binarias de género (hombre, mujer, niña, etc.) basadas en el aspecto físico de una persona en una imagen determinada. Este tipo de predicción no está diseñado para categorizar la identidad de género de una persona y no debe utilizarse Amazon Rekognition para realizar esta determinación. Por ejemplo, un actor masculino que lleva una peluca de pelo largo y pendientes para un papel podría predecirse como mujer.

El uso de Amazon Rekognition para hacer predicciones binarias de género es más adecuado para casos de uso en los que las estadísticas agregadas de distribución de género deben analizarse 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 recomendamos utilizar predicciones binarias de género para tomar decisiones que afecten a los derechos, la privacidad o el acceso a los servicios de una persona.

Amazon Rekognition devuelve las etiquetas en inglés. Puede usar[Amazon Translate](#) traducir etiquetas inglesas a[otros idiomas](#).

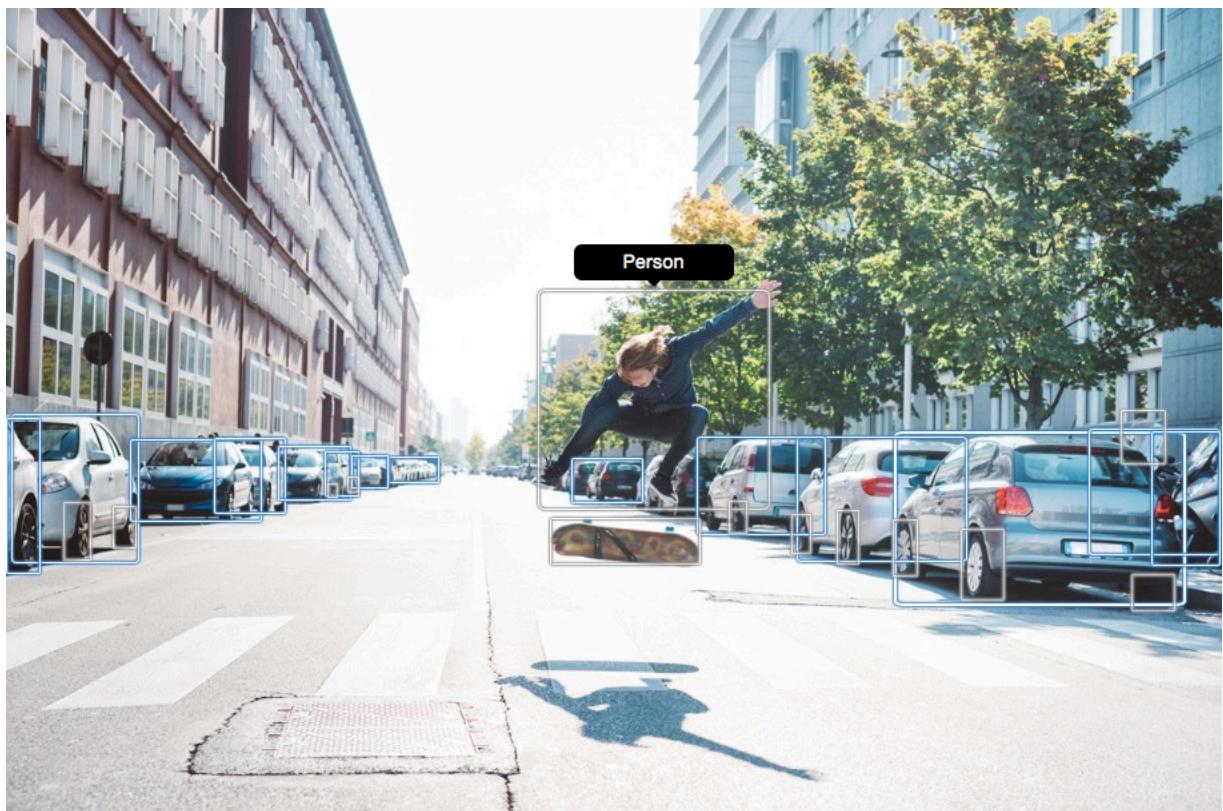
## Cuadros delimitadores y etiquetas padre

Amazon Rekognition Image y Amazon Rekognition Video pueden devolver el cuadro delimitador de etiquetas de objetos comunes como 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.

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.

Amazon Rekognition Image y Amazon Rekognition Video devuelven la versión del modelo de detección de etiquetas utilizado para detectar etiquetas en una imagen o vídeo almacenado.

Por ejemplo, en la imagen siguiente, Amazon Rekognition Image es capaz de detectar la presencia de una persona, un patín, coches aparcados y otra información. Amazon Rekognition Image también devuelve el cuadro delimitador de una persona detectada y otros objetos detectados, como automóviles y ruedas. 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.



## Detección de etiquetas en una imagen

Puede utilizar la operación de [DetectLabels \(p. 583\)](#) para detectar etiquetas en una imagen. Para ver un ejemplo, consulte [Análisis de imágenes almacenadas en un bucket de Amazon S3 \(p. 30\)](#).

En los siguientes ejemplos se emplean varios AWS SDK y la AWS CLI para llamar a `DetectLabels`. Para obtener más información sobre la respuesta de la operación `DetectLabels`, consulte [Respuesta DetectLabels \(p. 143\)](#).

Para detectar las etiquetas en una imagen

1. Si aún no lo ha hecho:
  - a. Crear o actualizar un usuario de IAM con `AmazonRekognitionFullAccess` y `AmazonS3ReadOnlyAccess` permisos. Para obtener más información, consulte [Paso 1: Configurar una cuenta de AWS y crear un usuario de IAM \(p. 13\)](#).
  - b. Instale y configure la AWS CLI y los AWS SDK. Para obtener más información, consulte [Paso 2: Configurar la AWS CLI y los SDK de \(p. 14\)](#).
2. Cargue una imagen que contenga uno o varios objetos (árboles, casas, barcos) en el bucket de S3. La imagen debe estar en formato .jpg o .png.

Para obtener 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 la operación `DetectLabels`.

Java

Este ejemplo muestra una lista de las etiquetas que se han detectado en la imagen de entrada. Sustituir los valores debucket y photo con los nombres del bucket de Amazon S3 e 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. Sustituir los valores `debucketyphoto` con los nombres del bucket de Amazon S3 e imagen que utilizó en el paso 2.

```
aws rekognition detect-labels \
--image '{"S3Object":{"Bucket": "bucket", "Name": "file"}}'
```

#### Python

Este ejemplo muestra las etiquetas que se han detectado en la imagen de entrada. En la función `main`, sustituya los valores `debucketyphoto` con los nombres del bucket de Amazon S3 e 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.)

import boto3

def detect_labels(photo, bucket):

 client=boto3.client('rekognition')

 response = client.detect_labels(Image={'S3Object':
{'Bucket':bucket,'Name':photo}},
 MaxLabels=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 ("-----")
 print ()
 return len(response['Labels'])

def main():
 photo=''
 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. Sustituir los valores de `bucket` y `photo` con los nombres del bucket de Amazon S3 e 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 = "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. Sustituir los valores debucket y photo con los nombres del bucket de Amazon S3 e imagen que utilizó en el paso 2.

```
#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.)

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. Sustituir los valores debucket y photo con los nombres del bucket de Amazon S3 e imagen que utilizó en el paso 2.

Si utiliza definiciones de TypeScript, es posible que deba usar `import AWS from 'aws-sdk'` en lugar de `const AWS = require('aws-sdk')`, para ejecutar el programa con Node.js. Puede consultar el [AWSSDK para Javascript](#) para obtener más información. Según cómo haya configurado las configuraciones, es posible que también tenga que especificar su región con `AWS.config.update({region:region});`.

```
//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.)

// Load the SDK
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,
 region: process.env.AWS_REGION
})
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 toma desde el AWSEjemplos de SDK de documentación Repositorio de GitHub.  
Ver el ejemplo completo[aquí](#).

```
public static void detectImageLabels(RekognitionClient rekClient, String sourceImage) {

 try {

 InputStream sourceStream = new URL("https://images.unsplash.com/photo-1557456170-0cf4f4d0d362?ixid=MnwxMjA3fDB8MHxzZWfY2h8MXx8bGFrZXxlbnwwfHwwfHw%3D&ixlib=rb-1.2.1&w=1000&q=80").openStream();
 // InputStream sourceStream = new FileInputStream(sourceImage);
 SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);

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

## Solicitud de operación DetectLabels

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.

```
{
 "Image": {
 "S3Object": {
 "Bucket": "bucket",
 "Name": "input.jpg"
 }
 }
}
```

```
 },
 "MaxLabels": 10,
 "MinConfidence": 75
}
```

## Respuesta DetectLabels

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`.

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.

```
{
 {
 "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"
 }
]
 },
],
 "LabelModelVersion": 1
 }
}
```

```
{
 "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"
}

}
```

## Detección de etiquetas en un vídeo

Amazon Rekognition Video puede detectar etiquetas y el momento en el 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 \(SDK\) \(p. 72\)](#). Para ver un ejemplo de AWS CLI, consulte [Análisis de un vídeo con la AWS Command Line Interface \(p. 89\)](#).

La detección de etiquetas de Amazon Rekognition Video es una operación asíncrona. Para iniciar la detección de etiquetas en un vídeo, llame a [StartLabelDetection \(p. 701\)](#). Amazon Rekognition Video publica el estado de la realización del análisis de vídeo en un tema de Amazon Simple Notification Service. Si el análisis de vídeo es correcto, llame a [GetLabelDetection \(p. 626\)](#) para obtener las etiquetas detectadas. Para obtener información sobre cómo llamar a operaciones de la API de análisis de vídeo, consulte [Llamar a las operaciones de Amazon Rekognition Video \(p. 66\)](#).

## Respuesta de la operación GetLabelDetection

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 especificando el parámetro `SortBy`.

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 \(p. 794\)](#)).
- Información de etiqueta— El `LabelDetection` elemento array contiene un ([Label \(p. 793\)](#)) que 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, en milisegundos, transcurrido desde el comienzo del vídeo hasta que se detectó la etiqueta.
- 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 \(p. 68\)](#).
- Información de vídeo: la respuesta incluye información acerca del formato de vídeo (`VideoMetadata`) en cada página de información devuelta por GetLabelDetection.

```
{
 "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
}
}
```

## Detección de etiquetas personalizadas

Amazon Rekognition Custom Labels permite identificar los objetos y las escenas de las imágenes específicas de sus necesidades empresariales, como logotipos o piezas de máquinas de ingeniería. Para obtener más información, consulte [¿Qué son las etiquetas personalizadas de Amazon Rekognition?](#) en la Amazon Rekognition Custom Labels Development.

# Detección y análisis de rostros

Amazon Rekognition puede detectar rostros en imágenes y vídeos. En esta sección se explican las operaciones sin almacenamiento para análisis de rostros. Con Amazon Rekognition puede obtener información acerca de dónde se detectan rostros en una imagen o vídeo, referencias faciales, por ejemplo, la posición de los ojos y las emociones detectadas (por ejemplo, una apariencia de felicidad o de tristeza). También puede comparar un rostro en una imagen con los rostros detectados en otra imagen.

Cuando se proporciona una imagen que contiene un rostro, Amazon Rekognition 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.



En esta sección se proporcionan ejemplos de análisis de imágenes y análisis facial de vídeo. Para obtener más información sobre el uso de la API de Amazon Rekognition, consulte [Trabajar con imágenes \(p. 28\)](#) y [Trabajar con vídeos almacenados \(p. 63\)](#).

## Note

Los modelos de detección de rostros utilizados por Amazon Rekognition Image y Amazon Rekognition Video no admiten la detección de rostros de dibujos animados, personajes animados o entidades no humanas. Si desea detectar personajes de dibujos animados en imágenes o vídeos, le recomendamos que utilice etiquetas personalizadas de Amazon Rekognition. Para obtener más información, consulte la [Amazon Rekognition](#).

Puede utilizar las operaciones de almacenamiento 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 \(p. 181\)](#).

## Temas

- [Información general de la detección y comparación de rostros \(p. 150\)](#)
- [Directrices sobre los atributos faciales \(p. 150\)](#)
- [Detección de rostros en una imagen \(p. 151\)](#)
- [Comparación de rostros en imágenes \(p. 163\)](#)

- [Detección de rostros en un vídeo almacenado \(p. 172\)](#)

## Información general de la detección y comparación de rostros

Existen dos aplicaciones principales del aprendizaje automático que analizan las imágenes que contienen rostros: la detección y la comparación. Un sistema de detección facial está diseñado para responder a esta pregunta: ¿hay un rostro en esta imagen? Un sistema de detección facial determina la presencia, la ubicación, la escala y (posiblemente) la orientación de cualquier rostro presente en una fotografía o en un fotograma de video. Este sistema se ha diseñado para detectar la presencia de rostros independientemente de atributos como el género, la edad o el vello facial.

Los sistemas de comparación facial están diseñados para responder a esta pregunta: ¿el rostro de una imagen coincide con el rostro de otra imagen? Los sistemas de comparación facial toman la imagen de un rostro y predicen si ese rostro coincide con otros rostros de la base de datos especificada. Los sistemas de comparación facial están diseñados para comparar y predecir posibles coincidencias de rostros, independientemente de la expresión, el vello facial y la edad.

Tanto los sistemas de detección como los sistemas de comparación de rostros pueden ofrecer una estimación aproximada del nivel de confianza de la predicción en forma de probabilidad o puntuación de confianza. Por ejemplo, un sistema de detección facial puede predecir que una región de una imagen es un rostro con una puntuación del nivel de confianza del 90 % y que otra región de la imagen es un rostro con una puntuación de confianza del 60 %. Es más probable que el rostro esté contenido en la región con la mayor puntuación de confianza. Si un sistema de detección facial no detecta correctamente un rostro, o proporciona una predicción con un nivel de confianza bajo respecto a un rostro real, esto se denomina una detección omitida o un falso negativo. Si un sistema de detección facial predice de forma incorrecta la presencia de un rostro con un alto nivel de confianza, se trata de una falsa alarma o de un falso positivo. Del mismo modo, un sistema de comparación facial podría no hallar ninguna coincidencia entre dos rostros que pertenecen a la misma persona (detección perdida o falso negativo) o predecir de forma incorrecta que los rostros de dos personas diferentes pertenecen a la misma persona (falsa alarma o falso positivo).

Las puntuaciones de confianza son un componente esencial de los sistemas de detección y comparación de rostros. Estos sistemas realizan predicciones con un nivel de confianza de si un rostro está presente en una imagen o de si coincide con el rostro de otra imagen. Los usuarios de estos sistemas deben tener en cuenta los umbrales de puntuación de confianza o de parecido que proporciona el sistema al diseñar su aplicación y tomar decisiones basadas en los resultados que ofrece ese sistema. Por ejemplo, en una foto que se utiliza para identificar a familiares que se parecen, si el umbral de confianza se establece en un 80 %, entonces la aplicación devolverá coincidencias cuyas predicciones presenten un nivel de confianza de hasta el 80 %, pero no las que se sitúen por debajo de ese nivel. Este umbral puede ser aceptable porque el riesgo de una detección perdida o de una falsa alarma es bajo en este tipo de uso. Sin embargo, en los casos de uso cuyo riesgo de detección perdida o falsa alarma es superior, el sistema debe utilizar un nivel de confianza mayor. Se debe utilizar un umbral de confianza o parecido del 99 % en aquellas situaciones en que es importante obtener coincidencias faciales con un alto nivel de precisión. Para obtener más información sobre los umbrales de confianza recomendados, consulte [Búsqueda de rostros en una colección \(p. 181\)](#).

## Directrices sobre los atributos faciales

Amazon Rekognition devuelve un cuadro delimitador, referencias, calidad y postura para cada rostro que detecta. Amazon Rekognition también devuelve predicciones de emoción, sexo, edad y otros atributos para cada cara si el parámetro de los atributos se establece en `ALL` en la solicitud de API. Cada atributo o emoción tiene un valor y una puntuación de confianza. Por ejemplo, se podría predecir que un rostro determinado tiene el sexo "Femenino" con una puntuación de confianza del 85 % y la emoción "Feliz" con una puntuación de confianza del 90 %.

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 Amazon 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.

De igual modo, las predicciones de las expresiones emocionales se basan únicamente en la apariencia física del rostro de una persona. No indica el estado emocional interno real y no debe utilizarse Amazon Rekognition para realizar esta determinación. Por ejemplo, en una imagen, una persona podría fingir estar feliz a través del rostro, aunque realmente no estuviera experimentando felicidad.

Recomendamos utilizar un umbral del 99 % o más en aquellos casos de uso en que la precisión de clasificación podría tener algún impacto negativo en los sujetos de las imágenes. La única excepción es el rango de edades, donde Amazon Rekognition calcula la edad inferior y superior para la persona. En este caso, cuanto mayor es el rango de edades, la más baja será la confianza de esa predicción. A título aproximativo, puede utilizar el punto medio del rango de edades para calcular un valor único de edad del rostro detectado. La edad real no se corresponderá necesariamente con este número.

Uno de los mejores usos de estos atributos es la generación de estadísticas agregadas. Por ejemplo, atributos como la sonrisa, la postura y la nitidez pueden utilizarse para seleccionar automáticamente la mejor "imagen de perfil" en una aplicación de redes sociales. Otro caso de uso común consiste en calcular de forma anónima los datos demográficos de una muestra amplia utilizando los atributos de género y edad (por ejemplo, en eventos o en comercios minoristas).

Para obtener más información acerca de los atributos, consulte [FaceDetail \(p. 773\)](#).

## Detección de rostros en una imagen

Amazon Rekognition Image proporciona el [DetectFaces \(p. 578\)](#) 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 los 100 rostros de mayor tamaño en 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 cargará una imagen (JPEG o PNG) en su bucket de S3; y especificará el nombre de clave del objeto.

Para detectar rostros en una imagen

1. Si aún no lo ha hecho:
  - a. Crear o actualizar un usuario de IAM con `AmazonRekognitionFullAccess` y `AmazonS3ReadOnlyAccess` permisos. Para obtener más información, consulte [Paso 1: Configurar una cuenta de AWS y crear un usuario de IAM \(p. 13\)](#).
  - b. Instale y configure la AWS CLI y los AWS SDK. Para obtener más información, consulte [Paso 2: Configurar la AWS CLI y AWSSDK de \(p. 14\)](#).
2. Cargue una imagen (que contenga uno o varios rostros) en el bucket de S3.  
Para obtener 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 fechas 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 `debucket` al 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(face));
 }
 } catch (AmazonRekognitionException e) {
 e.printStackTrace();
 }
 }
}
```

```
 }

}
```

## Java V2

Este código se toma desde el AWS Repository de GitHub de ejemplos del SDK de documentación. Ver el ejemplo completo [aquí](#).

```
public static void detectFacesinImage(RekognitionClient rekClient, String sourceImage) {

 try {
 InputStream sourceStream = new FileInputStream(new File(sourceImage));
 SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);

 // Create an Image object for the source image.
 Image souImage = Image.builder()
 .bytes(sourceBytes)
 .build();

 DetectFacesRequest facesRequest = DetectFacesRequest.builder()
 .attributes(Attribute.ALL)
 .image(souImage)
 .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 | FileNotFoundException e) {
 System.out.println(e.getMessage());
 System.exit(1);
 }
}
```

## AWS CLI

Este ejemplo muestra la salida de JSON de la operación `detect-faces` de la AWS CLI. Reemplace `file` por el nombre de un archivo de imagen. Reemplazar `bucket` con el nombre del bucket de Amazon S3 que contiene el archivo de imagen.

```
aws rekognition detect-faces \
--image '{"S3Object":{"Bucket":"'bucket", "Name":"'file"{}"}' \
--attributes "ALL"
```

## Python

Este ejemplo muestra el rango de fechas 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 bucket al 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.)

import boto3
import json

def detect_faces(photo, bucket):

 client=boto3.client('rekognition')

 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("Emotions: " + str(faceDetail['Emotions'][0]))

 return len(response['FaceDetails'])
def main():
 photo='photo'
 bucket='bucket'
 face_count=detect_faces(photo, bucket)
 print("Faces detected: " + str(face_count))

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

## .NET

Este ejemplo muestra el rango de fechas 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 bucket al 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 = "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 `bucket` por el nombre del 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.)

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 " mouth_open.value: #{face_detail.mouth_open.value}"
 puts " mouth_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 debucket al bucket de Amazon S3 donde se almacena la imagen.

Si utiliza definiciones de TypeScript, es posible que deba utilizar `import AWS from 'aws-sdk'` en lugar de `const AWS = require('aws-sdk')`, para ejecutar el programa con Node.js. Puede consultar el [AWS SDK para Javascript](#) para obtener más información. Según cómo haya configurado las configuraciones, es posible que también tenga que especificar su región con `AWS.config.update({region:region});`.

```
//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.)

const AWS = require('aws-sdk')
const bucket = 'bucket' // the bucketname without s3://
const photo = 'input.jpg' // the name of file
const config = new AWS.Config({
 accessKeyId: process.env.AWS_ACCESS_KEY_ID,
 secretAccessKey: process.env.AWS_SECRET_ACCESS_KEY,
 region: process.env.AWS_REGION
})
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
});
```

## Solicitud de operación DetectFaces

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 \(p. 28\)](#).

```
{
 "Image": {
 "S3Object": {
 "Bucket": "bucket",
 "Name": "input.jpg"
 }
 },
 "Attributes": [
 "ALL"
]
}
```

## Respuesta de la operación DetectFaces

`DetectFaces` devuelve la siguiente información por cada rostro detectado:

- **Bounding Box:** las coordenadas del cuadro delimitador que rodea el rostro.
- **Confianza:** grado de confianza de que el cuadro delimitador contiene un rostro.
- **Referentes 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.
- **Características faciales—** Un conjunto de atributos faciales, como si el rostro tiene barba. Para cada atributo, la respuesta proporciona un valor. 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.
- **Calidad:** describe el brillo y la nitidez del rostro. Para obtener información acerca de cómo optimizar la detección de rostros, consulte [Recomendaciones para imágenes de entrada de comparación facial \(p. 131\)](#).
- **Postura:** describe la rotación del rostro dentro de la imagen.

- Emociones: un conjunto de emociones con confianza en el análisis.

A continuación se muestra un ejemplo de respuesta de una llamada a la API DetectFaces.

```
{
 "FaceDetails": [
 {
 "AgeRange": {
 "High": 43,
 "Low": 26
 },
 "Beard": {
 "Confidence": 97.48941802978516,
 "Value": true
 },
 "BoundingBox": {
 "Height": 0.6968063116073608,
 "Left": 0.26937249302864075,
 "Top": 0.11424895375967026,
 "Width": 0.42325547337532043
 },
 "Confidence": 99.99995422363281,
 "Emotions": [
 {
 "Confidence": 0.042965151369571686,
 "Type": "DISGUSTED"
 },
 {
 "Confidence": 0.002022328320890665,
 "Type": "HAPPY"
 },
 {
 "Confidence": 0.4482877850532532,
 "Type": "SURPRISED"
 },
 {
 "Confidence": 0.007082826923578978,
 "Type": "ANGRY"
 },
 {
 "Confidence": 0,
 "Type": "CONFUSED"
 },
 {
 "Confidence": 99.47616577148438,
 "Type": "CALM"
 },
 {
 "Confidence": 0.017732391133904457,
 "Type": "SAD"
 }
],
 "Eyeglasses": {
 "Confidence": 99.42405700683594,
 "Value": false
 },
 "EyesOpen": {
 "Confidence": 99.99604797363281,
 "Value": true
 },
 "Gender": {
 "Confidence": 99.722412109375,
 "Value": "Male"
 },
 "Race": {
 "Confidence": 99.99995422363281,
 "Value": "White"
 }
 }
]
}
```

```
"Landmarks": [
 {
 "Type": "eyeLeft",
 "X": 0.38549351692199707,
 "Y": 0.3959200084209442
 },
 {
 "Type": "eyeRight",
 "X": 0.5773905515670776,
 "Y": 0.394561767578125
 },
 {
 "Type": "mouthLeft",
 "X": 0.40410104393959045,
 "Y": 0.6479480862617493
 },
 {
 "Type": "mouthRight",
 "X": 0.5623446702957153,
 "Y": 0.647117555141449
 },
 {
 "Type": "nose",
 "X": 0.47763553261756897,
 "Y": 0.5337067246437073
 },
 {
 "Type": "leftEyeBrowLeft",
 "X": 0.3114689588546753,
 "Y": 0.3376390337944031
 },
 {
 "Type": "leftEyeBrowRight",
 "X": 0.4224424660205841,
 "Y": 0.3232649564743042
 },
 {
 "Type": "leftEyeBrowUp",
 "X": 0.36654090881347656,
 "Y": 0.3104579746723175
 },
 {
 "Type": "rightEyeBrowLeft",
 "X": 0.5353175401687622,
 "Y": 0.3223199248313904
 },
 {
 "Type": "rightEyeBrowRight",
 "X": 0.6546239852905273,
 "Y": 0.3348073363304138
 },
 {
 "Type": "rightEyeBrowUp",
 "X": 0.5936762094497681,
 "Y": 0.3080498278141022
 },
 {
 "Type": "leftEyeLeft",
 "X": 0.3524211347103119,
 "Y": 0.3936865031719208
 },
 {
 "Type": "leftEyeRight",
 "X": 0.4229775369167328,
 "Y": 0.3973258435726166
},
```

```
{
 "Type": "leftEyeUp",
 "X": 0.38467878103256226,
 "Y": 0.3836822807788849
,
 {
 "Type": "leftEyeDown",
 "X": 0.38629674911499023,
 "Y": 0.40618783235549927
,
 {
 "Type": "rightEyeLeft",
 "X": 0.5374732613563538,
 "Y": 0.39637991786003113
,
 {
 "Type": "rightEyeRight",
 "X": 0.609208345413208,
 "Y": 0.391626238822937
,
 {
 "Type": "rightEyeUp",
 "X": 0.5750962495803833,
 "Y": 0.3821527063846588
,
 {
 "Type": "rightEyeDown",
 "X": 0.5740782618522644,
 "Y": 0.40471214056015015
,
 {
 "Type": "noseLeft",
 "X": 0.4441811740398407,
 "Y": 0.5608476400375366
,
 {
 "Type": "noseRight",
 "X": 0.5155643820762634,
 "Y": 0.5569332242012024
,
 {
 "Type": "mouthUp",
 "X": 0.47968366742134094,
 "Y": 0.6176465749740601
,
 {
 "Type": "mouthDown",
 "X": 0.4807897210121155,
 "Y": 0.690782368183136
,
 {
 "Type": "leftPupil",
 "X": 0.38549351692199707,
 "Y": 0.3959200084209442
,
 {
 "Type": "rightPupil",
 "X": 0.5773905515670776,
 "Y": 0.394561767578125
,
 {
 "Type": "upperJawlineLeft",
 "X": 0.27245330810546875,
 "Y": 0.3902156949043274
,
```

```
 "Type": "midJawlineLeft",
 "X": 0.31561678647994995,
 "Y": 0.6596118807792664
 },
 {
 "Type": "chinBottom",
 "X": 0.48385748267173767,
 "Y": 0.8160444498062134
 },
 {
 "Type": "midJawlineRight",
 "X": 0.6625112891197205,
 "Y": 0.656606137752533
 },
 {
 "Type": "upperJawlineRight",
 "X": 0.7042999863624573,
 "Y": 0.3863988518714905
 }
],
"MouthOpen": {
 "Confidence": 99.83820343017578,
 "Value": false
},
"Mustache": {
 "Confidence": 72.20288848876953,
 "Value": false
},
"Pose": {
 "Pitch": -4.970901966094971,
 "Roll": -1.4911699295043945,
 "Yaw": -10.983647346496582
},
"Quality": {
 "Brightness": 73.81391906738281,
 "Sharpness": 86.86019134521484
},
"Smile": {
 "Confidence": 99.93638610839844,
 "Value": false
},
"Sunglasses": {
 "Confidence": 99.81478881835938,
 "Value": false
}
}
]
}
```

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 devuelven son: `eyeLeft`, `eyeRight`, `nose`, `mouthLeft` y `mouthRight`.

# Comparación de rostros en imágenes

Para comparar un rostro en la imagen de origen con cada rostro en la imagen de destino, utilice la operación [CompareFaces \(p. 515\)](#).

Para especificar el nivel mínimo de confianza en la coincidencia que desea que se devuelva en la respuesta, utilice `similarityThreshold` en la solicitud. Para obtener más información, consulte [CompareFaces \(p. 515\)](#).

Si proporciona una imagen de origen que contenga varios rostros, el servicio detecta el rostro mayor y lo usa para compararlo con cada rostro detectado en la imagen de destino.

Puede proporcionar las imágenes de origen y de destino como una matriz de bytes de imagen (bytes de imagen cifrados en base64) o especificar objetos de Amazon S3. En el navegador AWS CLI ejemplo, cargará dos imágenes JPEG en su bucket de Amazon S3 y especificará el nombre de clave del objeto. En los demás ejemplos, puede cargar dos archivos desde el sistema de archivos local e introducirlos como una matriz de bytes de imagen.

## Note

CompareFaces utiliza algoritmos de aprendizaje automático, que son probabilísticos. Un falso negativo es una predicción incorrecta de que una cara de la imagen de destino tiene una puntuación de confianza de similitud baja en comparación con la cara de la imagen de origen. Para reducir la probabilidad de falsos negativos, recomendamos comparar la imagen de destino con varias imágenes de origen. Si planea utilizar CompareFaces para tomar una decisión que afecte a los derechos, la privacidad o el acceso a los servicios de una persona, te recomendamos que pases el resultado a un humano para que lo revise y lo valide antes de tomar medidas.

## Para comparar rostros

1. Si aún no lo ha hecho:
  - a. Crear o actualizar un usuario de IAM con `AmazonRekognitionFullAccess` y `AmazonS3ReadOnlyAccess` (AWS CLI solo ejemplo) permisos. Para obtener más información, consulte [Paso 1: Configurar una cuenta de AWS y crear un usuario de IAM \(p. 13\)](#).
  - b. Instale y configure la AWS CLI y los AWS SDK. Para obtener más información, consulte [Paso 2: Configurar la AWS CLI y AWSSDK de \(p. 14\)](#).
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> uncompered = compareFacesResult.getUnmatchedFaces();

 System.out.println("There was " + uncompered.size()
 + " face(s) that did not match");
 }
}
```

## Java V2

Este código se toma desde el AWS Repository de GitHub de ejemplos del SDK de documentación.  
Ver el ejemplo completo[aquí](#).

```
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> uncompered = compareFacesResult.unmatchedFaces();
 System.out.println("There was " + uncompered.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);
 }
}
```

## AWS CLI

Este ejemplo muestra la salida de JSON de la operación `compare-faces` de la AWS CLI.

Reemplazar `bucket-name` con 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 \
--source-image '{"S3Object":{"Bucket":"bucket-name","Name":"source.jpg"}' \
--target-image '{"S3Object":{"Bucket":"bucket-name","Name":"target.jpg"}'
```

## 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.

```
#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):

 client=boto3.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'
 target_file='target'
 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.

```
//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.)

const AWS = require('aws-sdk')
const bucket = 'bucket' // the bucketname without s3://
const photo_source = 'source.jpg'
const photo_target = 'target.jpg'
const config = new AWS.Config({
 accessKeyId: process.env.AWS_ACCESS_KEY_ID,
 secretAccessKey: process.env.AWS_SECRET_ACCESS_KEY,
 region: process.env.AWS_REGION
})
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
});
```

## Solicitud de operación CompareFaces

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 \(p. 28\)](#).

```
{
 "SourceImage": {
 "Bytes": "/9j/4AAQSk2Q==..."
 },
 "TargetImage": {
 "Bytes": "/9j/401Q==..."
 },
 "SimilarityThreshold": 70
}
```

## Respuesta de la operación CompareFaces

En la respuesta, obtiene una matriz de coincidencias de rostros, información de rostros de origen, orientación de imagen de origen y de destino y una matriz de rostros sin coincidencias. Para cada rostro coincidente de la imagen de destino, la respuesta proporciona una puntuación de similitud (cuánto se parece el rostro al rostro de origen) y los metadatos del rostro. Los metadatos de los rostros incluye información como el cuadro delimitador de la cara coincidente y una matriz de referencias faciales. La matriz de caras sin correspondencia incluye metadatos de rostro.

En la siguiente respuesta de ejemplo, tenga en cuenta lo siguiente:

- Información de coincidencia facial: el ejemplo muestra la coincidencia de rostro encontrada en la imagen de destino. Para esa coincidencia de rostro, proporciona un cuadro delimitador y un valor de confianza (el nivel de confianza de que tiene Amazon Rekognition de que el contorno contenga un rostro). La puntuación de `similarity` de 99.99 indica qué similitud guardan los rostros. La información de coincidencia de rostros incluye además una gama de ubicaciones de referencias.

Si varios rostros coinciden, la matriz `FaceMatches` incluye todas las coincidencias de rostros.

- Información del rostro de origen: la respuesta incluye información sobre el rostro de la imagen de origen que se utilizó para la comparación, incluido el cuadro delimitador y el valor de confianza.
- Información de coincidencia facial inigualable— El ejemplo muestra un rostro que ha encontrado Amazon Rekognition en la imagen de destino que no coincidía con el rostro analizado en la imagen de origen. Para ese rostro, proporciona un cuadro delimitador y un valor de confianza que indica el nivel de confianza de que tiene Amazon Rekognition de que el contorno contenga un rostro. La información de rostro incluye además una gama de ubicaciones de referencias.

Si Amazon Rekognition encuentra varios rostros que no coinciden, la `UnmatchedFacesarray` incluye todos los rostros que no coinciden.

```
{
 "FaceMatches": [{
 "Face": {
 "BoundingBox": {
 "Width": 0.5521978139877319,
 "Top": 0.1203877404332161,
 "Left": 0.23626373708248138,
 "Height": 0.3126954436302185
 },
 "Confidence": 99.98751068115234,
 "UnmatchedFace": {
 "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": 0.48565614223480225,
 "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 como se explica en el[the section called “Directrices sobre los atributos faciales” \(p. 150\)](#)(Se ha creado el certificado).

La detección de rostros de Amazon Rekognition Video en vídeos almacenados una operación asíncrona. Para iniciar la detección de rostros en vídeos llame a[StartFaceDetection \(p. 693\)](#). Amazon Rekognition Video publica el estado de realización del análisis de vídeo en un tema de Amazon Simple Notification Service (Amazon SNS). Si el análisis de vídeo es correcto, puede llamar a [GetFaceDetection \(p. 615\)](#) para obtener los resultados del análisis de vídeo. Para obtener más información sobre cómo iniciar el análisis de vídeo y obtener los resultados, consulte [Llamar a las operaciones de Amazon Rekognition Video \(p. 66\)](#).

Este procedimiento se expande en el código de[Análisis de un vídeo almacenado en un bucket de Amazon S3 con Java o Python \(SDK\) \(p. 72\)](#), 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 \(SDK\) \(p. 72\)](#).
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.)

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(bucket, video);

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

por:

```
StartFaceDetection(bucket, video);

if (GetSQSMessageSuccess() == true)
 GetFaceDetectionResults();
```

## Java V2

Este código se toma desde el AWS Repository de GitHub de ejemplos del SDK de documentación. Ver el ejemplo completo[aquí](#).

```
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 beard = face.face().beard().toString();
 String eyeglasses = face.face().eyeglasses().toString();
 String eyesOpen = face.face().eyesOpen().toString();
 String mustache = face.face().mustache().toString();
 String smile = face.face().smile().toString();
 }

 } 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.)

===== 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 \(SDK\) \(p. 72\)](#), el nombre de la función que se va a reemplazar es distinto.

- Ejecute el código. Se muestra información sobre los rostros detectados en el vídeo.

## Respuesta de la operación GetFaceDetection

GetFaceDetection devuelve una matriz (`Faces`) que contiene información sobre los rostros detectados en el vídeo. Un elemento de matriz, [FaceDetection \(p. 776\)](#), 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:

- Información de rostro: el elemento de matriz `FaceDetection` contiene información sobre el rostro detectado ([FaceDetail \(p. 773\)](#)) y el momento en que se detectó en el vídeo (`Timestamp`).
- 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 \(p. 68\)](#).
- Información de vídeo: la respuesta incluye información acerca del formato de vídeo (`VideoMetadata`) de cada página de información devuelta por GetFaceDetection.

```
{
 "Faces": [
 {
 "Face": {
 "BoundingBox": {
 "Height": 0.2300000417232513,
 "Left": 0.42500001192092896,
 "Top": 0.1633332657814026,
 "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": "mouth",
 "X": 0.5068183541297913,
 "Y": 0.23705792427062988
 }
]
 }
 }
]
}
```

```
 "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": 0.219999998079071,
 "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 puede almacenar información sobre rostros detectados en contenedores del servidor conocidos como colecciones. Puede utilizar la información facial almacenada en una colección para buscar rostros conocidos en imágenes, vídeos almacenados y vídeos en streaming. Amazon Rekognition admite la[IndexFaces](#) (p. 644). 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 conserva la información en el servidor.

Para poder almacenar información facial, primero debe crear una colección de rostros ([CreateCollection](#) (p. 522)) 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. Para buscar rostros en una imagen, llame a [SearchFacesByImage](#) (p. 680). Para buscar rostros en un vídeo almacenado, llame a [StartFaceSearch](#) (p. 697). Para buscar rostros en un vídeo en streaming, llame a [CreateStreamProcessor](#) (p. 537).

## Note

El servicio no conserva bytes de imágenes reales. En lugar de ello, el algoritmo de detección subyacente detecta primero los rostros en la imagen de entrada, extrae los rasgos faciales en un vector de rasgos de cada rostro y después los almacena en la colección. Amazon Rekognition utiliza estos vectores de rasgos cuando busca rostros coincidentes.

Puede utilizar las colecciones en diversas situaciones. Por ejemplo, puede crear una colección de rostros para almacenar imágenes de credenciales escaneadas utilizando la operación [IndexFaces](#). 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.

## Administración de colecciones

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. Las colecciones de rostros se crean en unAWSRegión de la 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 [Control de versiones del modelo](#) (p. 10).

Puede realizar las siguientes operaciones de administración en una colección.

- Crear una colección con [the section called “CreateCollection”](#) (p. 522). Para obtener más información, consulte [Creación de una colección](#) (p. 185).
- Enumerar las colecciones disponibles con [the section called “ListCollections”](#) (p. 653). Para obtener más información, consulte [Listado de colecciones](#) (p. 192).
- Describir una colección con [the section called “DescribeCollection”](#) (p. 556). Para obtener más información, consulte [Descripción de una colección](#) (p. 196).
- Eliminar una colección con [the section called “DeleteCollection”](#) (p. 541). Para obtener más información, consulte [Eliminación de una colección](#) (p. 201).

## 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 [the section called “IndexFaces” \(p. 644\)](#) detecta rostros en la imagen de entrada (JPEG o PNG) y las añade a la colección de rostros especificada. Se devuelve un ID único de rostro por cada rostro detectado en la imagen. Una vez guardados los rostros, puede buscar rostros coincidentes en la colección. Para obtener más información, consulte [Incorporación de rostros en una colección \(p. 204\)](#).
- La operación [the section called “ListFaces” \(p. 663\)](#) enumera los rostros de una colección. Para obtener más información, consulte [Incorporación de rostros en una colección \(p. 204\)](#).
- La operación [the section called “DeleteFaces” \(p. 545\)](#) elimina los rostros de una colección. Para obtener más información, consulte [Eliminación de rostros de una colección \(p. 221\)](#).

## Directrices para usar IndexFaces

A continuación se presentan las directrices para usar `IndexFaces` en situaciones comunes.

### Aplicaciones críticas o de seguridad pública

- Llame a [IndexFaces \(p. 644\)](#) 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 \(p. 578\)](#) 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 para compartir fotos 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.) 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 imágenes de entrada de comparación facial \(p. 131\)](#).

## Búsqueda de caras dentro de una colección

Después de crear una colección de rostros y almacenar rostros, puede buscar rostros coincidentes en una colección de rostros. Con Amazon Rekognition, puede buscar rostros en una colección que coincidan con:

- Un ID de rostro proporcionado ([the section called “SearchFaces” \(p. 676\)](#)). Para obtener más información, consulte [Búsqueda de un rostro mediante su ID de cara \(p. 224\)](#).
- El rostro de mayor tamaño de una imagen proporcionada ([the section called “SearchFacesByImage” \(p. 680\)](#)). Para obtener más información, consulte [Búsqueda de un rostro mediante una imagen \(p. 229\)](#).
- Rostros en un vídeo almacenado. Para obtener más información, consulte [Búsqueda de rostros en vídeos almacenados \(p. 234\)](#).
- Rostros en un vídeo en streaming. Para obtener más información, consulte [Uso de videos en streaming \(p. 94\)](#).

La operación `CompareFaces` y las operaciones de búsqueda de rostros tienen las siguientes diferencias:

- La operación `CompareFaces` compara 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 \(p. 163\)](#).
- `SearchFaces` y `SearchFacesByImage` comparan un rostro (identificado por `FaceId` o una imagen de entrada) con todos los rostros de una colección de rostros especificada. Por tanto, el ámbito de esta búsqueda es mucho mayor. Además, como se conserva información de rasgos faciales de los rostros que ya están almacenados en la colección de rostros, puede buscar rostros coincidentes varias veces.

## Uso de umbrales de similitud para hacer coincidir caras

Le permitimos controlar los resultados de todas las operaciones de búsqueda ([CompareFaces \(p. 515\)](#), [SearchFaces \(p. 676\)](#) y [SearchFacesByImage \(p. 680\)](#)) proporcionando un umbral de similitud como parámetro de entrada.

El atributo de entrada del umbral de similitud para `SearchFaces` y `SearchFacesByImage`, `FaceMatchThreshold` controla la cantidad de resultados que se devuelven en función de la similitud con el rostro que se hace coincidir. (Este atributo 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 recuperación de los resultados de la búsqueda: mientras más bajo sea el umbral, mayor será la retirada de la.

Todos los sistemas de aprendizaje automático 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`, 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 `Similarity` (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.

## Casos de uso que implican 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, imágenes](#)

de entrada y videos (p. 131) y Directrices para usar IndexFaces (p. 182), 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 un 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 la [AWSCondiciones del servicio](#). 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 utilice 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 la comparación facial se utiliza en escenarios que no están relacionados con el cumplimiento de la ley (por ejemplo, para desbloquear un teléfono o autenticar la identidad de un empleado al acceder a un edificio de oficinas protegido y privado), estas decisiones no requieren una auditoría manual, ya que no afectan a las libertades civiles de los individuos o 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 ayudar a 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. Las respuestas de las operaciones de rostros de Amazon Rekognition le permiten obtener rápidamente un conjunto de posibles coincidencias para examinarlas más a fondo.
- No utilice las 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 sobre la base de su examen independiente 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 \(p. 644\)](#). Para obtener más información, consulte [Directrices para usar IndexFaces \(p. 182\)](#).

En otros casos de uso (por ejemplo, las redes sociales), recomendamos aplicar su propio sentido general para evaluar si los resultados de Amazon Rekognition necesitan 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 \(p. 522\)](#) para crear una colección.

Para obtener más información, consulte [Administración de colecciones \(p. 181\)](#).

Para crear una colección (SDK)

1. Si aún no lo ha hecho:

- a. Crear o actualizar un usuario de IAM con `AmazonRekognitionFullAccess` permisos. Para obtener más información, consulte [Paso 1: Configurar una cuenta de AWS y crear un usuario de IAM \(p. 13\)](#).

- b. Instale y configure la AWS CLI y los AWS SDK. Para obtener más información, consulte [Paso 2: Configurar la AWS CLI y el AWS SDK de \(p. 14\)](#).
2. Utilice 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 toma desde la AWS Documentación Ejemplos del SDK repositorio de GitHub. Ver el ejemplo completo [aquí](#).

```
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);
}
}
```

## AWS CLI

Este comando de la AWS CLI muestra la salida de JSON para la operación `create-collection` de la CLI.

Reemplace el valor de `collection-id` por el nombre de la colección que desea crear.

```
aws rekognition create-collection \
--collection-id "collectionname"
```

## 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.

```
#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):

 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='Collection'
 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` con el nombre de la región asociada a tu cuenta y reemplaza el valor de `collectionName` con el nombre deseado de tu colección.

```
import { CreateCollectionCommand} from "@aws-sdk/client-rekognition";
import { RekognitionClient } from "@aws-sdk/client-rekognition";

// Set the AWS Region.
const REGION = "region"; //e.g. "us-east-1"
const rekogClient = new RekognitionClient({ region: REGION });

// Name the collection
const collection_name = "collectionName"

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(collection_name)
```

## Solicitud de operación CreateCollection

La entrada de `CreateCollection` es el nombre de la colección que desea crear.

```
{
 "CollectionId": "MyCollection"
}
```

## Respuesta de la operación CreateCollection

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 identificar, organizar, buscar y filtrar 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 Identity and Access Management (IAM). Para obtener más información, consulte [Control del acceso a AWS Recursos que utilizan etiquetas de recursos](#).

### Temas

- [Incorporación de etiquetas en una colección nueva \(p. 189\)](#)
- [Incorporación de etiquetas en una colección existente \(p. 190\)](#)
- [Listar etiquetas de una colección \(p. 191\)](#)
- [Eliminar etiquetas de una colección \(p. 191\)](#)

## Incorporación de etiquetas en una colección nueva

También puede añadir etiquetas a una colección a medida que la crea mediante la `CreateCollection`. Especifique una o varias etiquetas en la `Tags` parámetro de entrada de matriz.

### AWS CLI

```
aws rekognition create-collection --collection-id "collection name"\
```

```
--tags '{"key1":"value1","key2":"value2"}'
```

### Python

```
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()
```

## Incorporación de etiquetas en una colección existente

Para añadir una o más etiquetas a una colección existente, utilice la `TagResource`. Especifique el nombre de recurso de Amazon (ARN) de la colección (`ResourceArn`) y las etiquetas (`Tags`) que desea agregar. En el ejemplo siguiente de la, se muestra cómo añadir dos etiquetas.

### AWS CLI

```
aws rekognition tag-resource --resource-arn resource-arn \
 --tags '{"key1":"value1","key2":"value2"}'
```

### Python

```
import boto3

def create_tag():
 client = boto3.client('rekognition')
 response = client.tag_resource(ResourceArn="arn:aws:rekognition:region-
name:5498347593847598:collection/NewCollectionName",
 Tags={
 "KeyName": "ValueName"
 })
 print(response)
 if "'HTTPStatusCode': 200" in str(response):
 print("Success!!")

def main():
 create_tag()

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

#### Note

Si no conoce el nombre de recurso de Amazon de la colección, puede utilizar `laDescribeCollection`.

## Listar etiquetas de una colección

Para enumerar las etiquetas asociadas a una colección, utilice la `ListTagsForResource` y especifique el ARN de la colección (`ResourceArn`). La respuesta es un mapa de las claves y los valores de las etiquetas que se asocian a la colección especificada.

#### AWS CLI

```
aws rekognition list-tags-for-resource --resource-arn resource-arn
```

#### Python

```
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()
```

La salida muestra una lista de etiquetas asociadas 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 `UntagResource`. Especifique el ARN del modelo (`ResourceArn`) y las teclas de etiqueta (`Tag-Keys`) que desea eliminar.

#### AWS CLI

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

Si lo desea, puede especificar las claves de las etiquetas en este formato:

```
--tag-keys key1,key2
```

## Python

```
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 \(p. 653\)](#) para obtener una lista de las colecciones de la región que utiliza.

Para obtener más información, consulte [Administración de colecciones \(p. 181\)](#).

### Para enumerar colecciones (SDK)

1. Si aún no lo ha hecho:
  - a. Crear o actualizar un usuario de IAM con `AmazonRekognitionFullAccess` permisos. Para obtener más información, consulte [Paso 1: Configurar una cuenta de AWS y crear un usuario de IAM \(p. 13\)](#).
  - b. Instale y configure la AWS CLI y los AWS SDK. Para obtener más información, consulte [Paso 2: Configurar la AWS CLI y AWSSDK de \(p. 14\)](#).
2. Utilice 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 toma desde la AWS Documentación Ejemplos del SDK repositorio de GitHub. Ver el ejemplo completo [aquí](#).

```
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);
 }
}
```

#### AWS CLI

Este comando de la AWS CLI muestra la salida de JSON para la operación list-collections de la CLI.

```
aws rekognition list-collections
```

## Python

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.)

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

```
import { ListCollectionsCommand } from "@aws-sdk/client-rekognition";
import { RekognitionClient } from "@aws-sdk/client-rekognition";

// Set the AWS Region.
const REGION = "region"; //e.g. "us-east-1"
const rekogClient = new RekognitionClient({ region: REGION });

const listCollection = async () => {
 var max_results = 3
 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
 });
 if (JSON.stringify(response).includes("NextToken")){
 nextToken = response.NextToken
 response = new ListCollectionsCommand({NextToken:nextToken, MaxResults:
max_results})
 }
 else{
 done=true
 }
 return collection_count
 }
}

var collect_list = await listCollection()
console.log(collect_list)
```

## Solicitud de operación ListCollections

La entrada de ListCollections es el número máximo de colecciones que se va a devolver.

```
{
 "MaxResults": 2
}
```

Si la respuesta tiene más colecciones de las solicitadas por 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
}
```

## Respuesta de operación ListCollections

Amazon Rekognition devuelve una matriz 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 [Control de versiones del modelo \(p. 10\)](#).

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 \(p. 556\)](#) 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 “Control de versiones del modelo” \(p. 10\)](#).
- 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. Crear o actualizar un usuario de IAM con `AmazonRekognitionFullAccess` permisos. Para obtener más información, consulte [Paso 1: Configurar una cuenta de AWS y crear un usuario de IAM \(p. 13\)](#).
  - b. Instale y configure la AWS CLI y los AWS SDK. Para obtener más información, consulte [Paso 2: Configurar la AWS CLI y el AWS SDK de \(p. 14\)](#).
2. Utilice 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 toma desde la AWS Documentación Ejemplos del SDK repositorio de GitHub. Ver el ejemplo completo [aquí](#).

```
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);
 }
}
```

## AWS CLI

Este comando de la AWS CLI muestra la salida de JSON para la operación `describe-collection` de la CLI. Cambie el valor de `collection-id` por el ID de la colección deseada.

```
aws rekognition describe-collection --collection-id collectionname
```

## Python

Este ejemplo describe una colección.

Cambie el valor `collection_id` 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.)

import boto3
from botocore.exceptions import ClientError

def describe_collection(collection_id):

 print('Attempting to describe collection ' + collection_id)
 client=boto3.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='MyCollection'
 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

```
import { DescribeCollectionCommand } from "@aws-sdk/client-rekognition";
import { RekognitionClient } from "@aws-sdk/client-rekognition";
import { stringify } from "querystring";

// Set the AWS Region.
const REGION = "region"; //e.g. "us-east-1"
const rekogClient = new RekognitionClient({ region: REGION });

// Name the collection
const collection_name = "collectionName"
const resourceArn = "resourceArn"

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)
```

## Solicitud de operación DescribeCollection

La entrada de `DescribeCollection` es el ID de la colección deseada, tal y como se muestra en el siguiente ejemplo de JSON.

```
{
 "CollectionId": "MyCollection"
}
```

## Respuesta de operación DescribeCollection

La respuesta incluye:

- El número de rostros que se indexan en la colección, `FaceCount`.
- La versión del modelo de rostros que se utiliza con la colección, `FaceModelVersion`. Para obtener más información, consulte [the section called “Control de versiones del modelo” \(p. 10\)](#).
- 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,
 "FaceModelVersion": "1.0"
}
```

## Eliminación de una colección

Puede utilizar la operación [DeleteCollection \(p. 541\)](#) para eliminar una colección.

Para obtener más información, consulte [Administración de colecciones \(p. 181\)](#).

Para eliminar una colección (SDK)

1. Si aún no lo ha hecho:
  - a. Crear o actualizar un usuario de IAM con `AmazonRekognitionFullAccess` permisos. Para obtener más información, consulte [Paso 1: Configurar una cuenta de AWS y crear un usuario de IAM \(p. 13\)](#).
  - b. Instale y configure la AWS CLI y los AWS SDK. Para obtener más información, consulte [Paso 2: Configurar la AWS CLI y el AWS SDK de \(p. 14\)](#).
2. Utilice 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 toma desde la AWS Documentación Ejemplos del SDK repositorio de GitHub. Ver el ejemplo completo [aquí](#).

```
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);
 }
}
```

## AWS CLI

Este comando de la AWS CLI muestra la salida de JSON para la operación `delete-collection` de la CLI. Reemplace el valor de `collection-id` por el nombre de la colección que desea eliminar.

```
aws rekognition delete-collection \
--collection-id "collectionname"
```

## Python

Este ejemplo elimina una colección.

Cambie el valor `collection_id` 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.)

import boto3
from botocore.exceptions import ClientError
from os import environ

def delete_collection(collection_id):
```

```
print('Attempting to delete collection ' + collection_id)
client=boto3.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='UnitTestCollection'
 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

```
import { DeleteCollectionCommand } from "@aws-sdk/client-rekognition";
import { RekognitionClient } from "@aws-sdk/client-rekognition";

// Set the AWS Region.
const REGION = "region"; //e.g. "us-east-1"
const rekogClient = new RekognitionClient({ region: REGION });

// Name the collection
const collection_name = "collectionName"

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)
```

## Solicitud de operación DeleteCollection

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"
}
```

## Respuesta de operación DeleteCollection

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}
```

## Incorporación de rostros en una colección

Puede utilizar la operación [IndexFaces \(p. 644\)](#) 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 imágenes de entrada de comparación facial \(p. 131\)](#).

Para cada rostro, la operación `IndexFaces` conserva la siguiente información:

- Características 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 (de que el cuadro delimitador contiene un rostro), los ID asignados por Amazon Rekognition (ID de rostro e ID de imagen) y un ID de imagen externo (si lo proporciona) 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:

- [the section called “ListFaces” \(p. 663\)](#)
- Operaciones de búsqueda de rostros: las respuestas de [the section called “SearchFaces” \(p. 676\)](#) y [the section called “SearchFacesByImage” \(p. 680\)](#) 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 [Control de versiones del modelo \(p. 10\)](#).

La información sobre los rostros indexados se devuelve en una matriz de objetos [the section called “FaceRecord” \(p. 778\)](#).

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 operación `IndexFaces` le permite filtrar los rostros que se indexan en 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 la barra de calidad que se utiliza para filtrar rostros (valor predeterminado).
- **LOW**— Todas las caras excepto las de menor calidad están indexadas.
- **MEDIUM**
- **HIGH**— Solo se indexan las caras de la más alta 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 [the section called “DescribeCollection” \(p. 556\)](#).

La información sobre los rostros no indexados por `IndexFaces` se devuelve en una matriz de objetos [the section called “UnindexedFace” \(p. 842\)](#). 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 \(p. 182\)](#).

#### Para agregar rostros a una colección (SDK)

1. Si aún no lo ha hecho:
  - a. Crear o actualizar un usuario de IAM conAmazonRekognitionFullAccessyAmazonS3ReadOnlyAccesspermisos. Para obtener más información, consulte [Paso 1: Configurar una cuenta de AWS y crear un usuario de IAM \(p. 13\)](#).
  - b. Instale y configure la AWS CLI y los AWS SDK. Para obtener más información, consulte [Paso 2: Configurar laAWS CLyAWSSDK de \(p. 14\)](#).
2. Cargue una imagen (que contenga uno o varios rostros) en su bucket de Amazon S3.  
Para obtener instrucciones, consulte [Carga de objetos en Amazon S3](#)en laAmazon Simple Storage Service.
3. Utilice los siguientes ejemplos para llamar a la operación `IndexFaces`.

#### Java

Este ejemplo muestra los identificadores 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. Sustituir los valores de `bucket` y `photo` con los nombres del bucket de Amazon S3 e 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 toma desde la AWS Documentación Ejemplos del SDK repositorio de GitHub. Ver el ejemplo completo [aquí](#).

```
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);

 // Display the results.
 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);
 }
}
```

## AWS CLI

Este comando de la AWS CLI muestra la salida de JSON para la operación `index-faces` de la CLI.

Reemplace el valor de `collection-id` por el nombre de la colección donde desea almacenar el rostro. Sustituir los valores `deBucketName` con el bucket de Amazon S3 y el 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.

```
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"
```

### Python

Este ejemplo muestra los identificadores 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. Sustituir los valores `debucketyphoto` con los nombres del bucket de Amazon S3 e 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.

```
#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):

 client=boto3.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='bucket'
 collection_id='collection'
 photo='photo'

 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 identificadores 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. Sustituir los valores `debucketyphoto` con los nombres del bucket de Amazon S3 e 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 = "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);
 }
}
```

## Solicitud de operación IndexFaces

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"
}
```

## Respuesta de la operación IndexFaces

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 el valor de la `MaxFaces` ha superado el parámetro de entrada: el parámetro `Reasons` array 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 [the section called “UnindexedFace” \(p. 842\)](#).

```
{
 "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-8fb9-9f187db2050d"
 },
 "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": 0.6034515215100001,
 "Y": 0.4010535478591919
 },
 {
 "Type": "mouth",
 "X": 0.5751981735229492,
 "Y": 0.4210535478591919
 }
]
 }
 }
]
}
```

```
 "Type": "nose",
 "X": 0.631452854480286,
 "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": 0.0877777850627899
 },
 "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)
- Nueve atributos faciales (gafas, barba, 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 [Control de versiones del modelo \(p. 10\)](#).

`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 [Obtener orientación de imagen y coordenadas de cuadro delimitador \(p. 57\)](#).

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

```
"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"
 }
],
"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": []
}
```

## Listado de rostros en una colección

Puede utilizar la operación [ListFaces \(p. 663\)](#) para obtener una lista de los rostros en una colección.

Para obtener más información, consulte [Administración de rostros en una colección \(p. 182\)](#).

Para mostrar una lista de los rostros de una colección (SDK)

1. Si aún no lo ha hecho:
  - a. Crear o actualizar un usuario de IAM con `AmazonRekognitionFullAccess` permisos. Para obtener más información, consulte [Paso 1: Configurar una cuenta de AWS y crear un usuario de IAM \(p. 13\)](#).
  - b. Instale y configure la AWS CLI y los AWS SDK. Para obtener más información, consulte [Paso 2: Configurar la AWS CLI y el AWS SDK de \(p. 14\)](#).
2. Utilice 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 toma desde la AWS Documentación Ejemplos del SDK repositorio de GitHub. Ver el ejemplo completo [aquí](#).

```
public static void listFacesCollection(RekognitionClient rekClient, String
collectionId) {

 try {
 ListFacesRequest facesRequest = ListFacesRequest.builder()
 .collectionId(collectionId)
 .maxResults(10)
 .build();

 ListFacesResponse facesResponse = rekClient.listFaces(facesRequest);

 // For each face in the collection, print out the confidence level and
 // face id value.
 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);
 }
}
```

## AWS CLI

Este comando de la AWS CLI muestra la salida de JSON para la operación `list-faces` de la CLI. Reemplace el valor de `collection-id` por el nombre de la colección que desea enumerar.

```
aws rekognition list-faces \
--collection-id "collection-id"
```

## Python

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.)

import boto3

def list_faces_in_collection(collection_id):
```

```
maxResults=2
faces_count=0
tokens=True

client=boto3.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'

 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));
}
```

## Solicitud de operación ListFaces

La entrada a `ListFaces` es el ID de la colección para la que desea mostrar una lista de rostros. `MaxResults` es el número máximo de rostros para devolver.

```
{
 "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
}
```

## Respuesta de la operación ListFaces

La respuesta de `ListFaces` es información acerca de los metadatos de los rostros almacenados en la colección especificada.

- `FaceModelVersion`: es la versión del modelo de rostros asociado a la colección. Para obtener más información, consulte [Control de versiones del modelo \(p. 10\)](#).
- `Rostros`— Información sobre los rostros de la colección. Esto incluye información acerca de [the section called “BoundingBox” \(p. 740\)](#), la confianza, los identificadores de imágenes y el ID de rostro. Para obtener más información, consulte [the section called “Face” \(p. 771\)](#).
- `NextToken`: token que se utiliza para obtener el siguiente conjunto de resultados.

```
{
 "FaceModelVersion": "3.0",
 "Faces": [
 {
 "BoundingBox": {
 "Height": 0.06333330273628235,
 "Left": 0.1718519926071167,
 "Top": 0.7366669774055481,
```

```
 "Width": 0.11061699688434601
 },
 "Confidence": 100,
 "ExternalImageId": "input.jpg",
 "FaceId": "0b683aed-a0f1-48b2-9b5e-139e9cc2a757",
 "ImageId": "9ba38e68-35b6-5509-9d2e-fcffa75d1653",
 "IndexFacesModelVersion": "5.0"
}
],
"NextToken": "sm+5yhtT3aeEVIR4WA...."
```

## Eliminación de rostros de una colección

Puede utilizar la operación [DeleteFaces \(p. 545\)](#) para eliminar rostros de una colección. Para obtener más información, consulte [Administración de rostros en una colección \(p. 182\)](#).

Para eliminar los rostros de una colección

1. Si aún no lo ha hecho:
  - a. Crear o actualizar un usuario de IAM con `AmazonRekognitionFullAccess` permisos. Para obtener más información, consulte [Paso 1: Configurar una cuenta de AWS y crear un usuario de IAM \(p. 13\)](#).
  - b. Instale y configure la AWS CLI y los AWS SDK. Para obtener más información, consulte [Paso 2: Configurar la AWS CLI y AWS SDK de \(p. 14\)](#).
2. Utilice 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 varios rostros, añada ID de rostro a la matriz `faces`.

```
//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 toma desde la AWS Documentación Ejemplos del SDK repositorio de GitHub. Ver el ejemplo completo [aquí](#).

```
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);
 }
}
```

## AWS CLI

Este comando de la AWS CLI muestra la salida de JSON para la operación `delete-faces` de la CLI. Reemplace el valor de `collection-id` por el nombre de la colección que contiene el rostro que desea eliminar. Reemplace el valor de `face-ids` por una matriz con los ID de los rostros que desea eliminar.

```
aws rekognition delete-faces --collection-id "collectionname" --face-ids
'["faceid"]'
```

## 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 varios rostros, añada ID de rostro a la matriz `faces`.

```
#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):

 client=boto3.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'
 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 varios rostros, añada ID de rostro a la lista `faces`.

```
//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);
}
}
```

## Solicitud de operación DeleteFaces

La entrada de DeleteFaces es el ID de la colección que contiene los rostros y una matriz con los ID de los rostros que se van a eliminar.

```
{
 "CollectionId": "MyCollection",
 "FaceIds": [
 "daf29cac-f910-41e9-851f-6eeb0e08f973"
]
}
```

## Respuesta de la operación DeleteFaces

La respuesta de DeleteFaces contiene una matriz de ID de los rostros que se eliminaron.

```
{
 "DeletedFaces": [
 "daf29cac-f910-41e9-851f-6eeb0e08f973"
]
}
```

## Búsqueda de un rostro mediante su ID de cara

Puede utilizar la operación [SearchFaces \(p. 676\)](#) para buscar rostros en una colección que coincidan con un ID de rostro suministrado.

El ID de rostro se devuelve en la respuesta de la operación [IndexFaces \(p. 644\)](#) cuando el rostro se detecta y se añade a una colección. Para obtener más información, consulte [Administración de rostros en una colección \(p. 182\)](#).

Para buscar un rostro en una colección con su ID de rostro (SDK)

1. Si aún no lo ha hecho:
  - a. Crear o actualizar un usuario de IAM con `AmazonRekognitionFullAccess` permisos. Para obtener más información, consulte [Paso 1: Configurar una cuenta de AWS y crear un usuario de IAM \(p. 13\)](#).
  - b. Instale y configure la AWS CLI y los AWS SDK. Para obtener más información, consulte [Paso 2: Configurar la AWS CLI y AWS SDK de \(p. 14\)](#).
2. Utilice 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 toma desde la AWS Documentación Ejemplos del SDK repositorio de GitHub. Ver el ejemplo completo [aquí](#).

```
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);

 // Display the results.
 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);
 }
}
```

## AWS CLI

Este comando de la AWS CLI muestra la salida de JSON para la operación `search-faces` de la 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.

```
aws rekognition search-faces \
--face-id face-id \
--collection-id "collection-id"
```

## 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.

```
#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
 client=boto3.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']) + "%")
 print
return len(face_matches)

def main():

 face_id='xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx'
 collection_id='MyCollection'

 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.

## Solicitud de operación SearchFaces

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": 70
}
```

## Respuesta de la SearchFaces

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 mediante una imagen

Puede utilizar la operación [SearchFacesByImage \(p. 680\)](#) 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 caras dentro de una colección \(p. 182\)](#).

Para buscar un rostro en una colección con una imagen (SDK)

1. Si aún no lo ha hecho:
  - a. Crear o actualizar un usuario de IAM con `AmazonRekognitionFullAccess` y `AmazonS3ReadOnlyAccess` permisos. Para obtener más información, consulte [Paso 1: Configurar una cuenta de AWS y crear un usuario de IAM \(p. 13\)](#).
  - b. Instale y configure la AWS CLI y los AWS SDK. Para obtener más información, consulte [Paso 2: Configurar la AWS CLI y el AWS SDK de \(p. 14\)](#).
2. Cargue una imagen (que contenga uno o varios rostros) en el bucket de S3.  
Para obtener instrucciones, consulte [Carga de objetos en Amazon S3](#) en la Amazon Simple Storage Service.
3. Utilice 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 toma desde la AWS Documentación Ejemplos del SDK repositorio de GitHub. Ver el ejemplo completo [aquí](#).

```
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) ;

// Display the results.
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);
}
}
```

## AWS CLI

Este comando de la AWS CLI muestra la salida de JSON para la operación `search-faces-by-image` de la 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.

```
aws rekognition search-faces-by-image \
--image '{"S3Object":{"Bucket": "bucket-name", "Name": "Example.jpg"} }' \
--collection-id "collection-id"
```

## 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: cambie el valor de `collectionId` a la colección que desea buscar y sustituya los valores de `bucket` y `photo` con los nombres del bucket de Amazon S3 e 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.)

import boto3

if __name__ == "__main__":
 bucket='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: cambie el valor de `collectionId` a la colección que desea buscar y sustituya los valores de `bucket` y `photo` con los nombres del bucket de Amazon S3 e 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 = "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);
 }
}
```

## Solicitud de operación SearchFacesByImage

Los parámetros de entrada de `SearchFacesByImage` 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": 70
}
```

## Respuesta de la operación SearchFacesByImage

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": 0.0633333253860474,
 "Left": 0.17185185849666595,
 "Top": 0.7366666793823242,
 "Width": 0.11061728745698929
},
"SearchedFaceConfidence": 99.99999237060547
}
```

## Búsqueda de rostros en vídeos almacenados

Puede buscar una colección de rostros que coincide 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 [Uso de vídeos en streaming \(p. 94\)](#).

Los rostros que busca se deben indexar primero en una colección utilizando [IndexFaces \(p. 644\)](#). Para obtener más información, consulte [Incorporación de rostros en una colección \(p. 204\)](#).

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 comenzar a buscar rostros en un vídeo almacenado, llame a [StartFaceSearch \(p. 697\)](#) y proporciona el ID de la colección que desea buscar. Amazon Rekognition Video publica el estado de la realización del análisis de vídeo en un tema de Amazon Simple Notification Service (Amazon SNS). Si el análisis de vídeo es correcto, llame a [GetFaceSearch \(p. 620\)](#) para obtener los resultados de la búsqueda. Para obtener más información sobre cómo iniciar el análisis de vídeo y obtener los resultados, consulte [Llamar a las operaciones de Amazon Rekognition Video \(p. 66\)](#).

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 se expande en el código de [Análisis de un vídeo almacenado en un bucket de Amazon S3 con Java o Python \(SDK\) \(p. 72\)](#), 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. Cree una colección ([p. 185](#)).
2. Indexe un rostro en la colección ([p. 204](#)).
3. Realice [Análisis de un vídeo almacenado en un bucket de Amazon S3 con Java o Python \(SDK\) \(p. 72\)](#).
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 toma desde la AWS Documentación Ejemplos del SDK repositorio de GitHub. Ver el ejemplo completo [aquí](#).

```
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 beard = face.face().beard().toString();
 String eyeglasses = face.face().eyeglasses().toString();
 String eyesOpen = face.face().eyesOpen().toString();
 String mustache = face.face().mustache().toString();
 String smile = face.face().smile().toString();
}

} 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 \(SDK\) \(p. 72\)](#), 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.

## Respuesta de la operación GetFaceSearch

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 \(p. 804\)](#), 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 \(p. 777\)](#), la información sobre la persona que coincide, [PersonDetail \(p. 802\)](#), y el momento en que se encontró una coincidencia de la persona en el vídeo.

```
{
 "JobStatus": "SUCCEEDED",
 "NextToken": "IJdbzkZfvBRqj8GPV82BPiZKkLOGCqDIgNZG/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-5ec5-90b8-6d190daaff4c3"
 },
 "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": 0.217777737379074,
"Left": 0.7593749761581421,
"Top": 0.13333334028720856,
"Width": 0.12250000238418579
},
"Face": {
 "BoundingBox": {
 "Height": 0.217777737379074,
 "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": 0.838888835906982,
 "Left": 0,
 "Top": 0.15833333134651184,
 "Width": 0.2369791716337204
},
 "Face": {
 "BoundingBox": {
 "Height": 0.20000000298023224,
 "Left": 0.029999999329447746,
```

```
 "Top": 0.219999988079071,
 "Width": 0.1124999701976776
 },
 "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
}
}
```

# Recorridos de las personas

Amazon Rekognition Video puede crear un seguimiento del recorrido que siguen las personas en los videos y proporcionar información como:

- La ubicación de las personas en un fotograma de video 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.

Las personas de Amazon Rekognition Video que realizan recorridos en los videos almacenados es una operación asíncrona. Para iniciar el recorrido de las personas en videos llame a [StartPersonTracking \(p. 705\)](#). Amazon Rekognition Video publica el estado de finalización del análisis de video en un tema de Amazon Simple Notification Service. Si el análisis de video es correcto, llame a [GetPersonTracking \(p. 630\)](#) para obtener los resultados del análisis de video. Para obtener más información sobre cómo llamar a las operaciones de la API de Amazon Rekognition Video, consulte [Llamar a las operaciones de Amazon Rekognition Video \(p. 66\)](#).

El siguiente procedimiento muestra cómo realizar un seguimiento de la ruta de las personas a través de un video almacenado en un bucket de Amazon S3. El ejemplo amplía el código de [Análisis de un vídeo almacenado en un bucket de Amazon S3 con Java o Python \(SDK\) \(p. 72\)](#) que utiliza una cola de Amazon Simple Queue Service para obtener el estado de realización de una solicitud de análisis de video.

Para detectar personas en un video 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 \(SDK\) \(p. 72\)](#).
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(bucket, video);

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

```

por:

```

StartPersonDetection(bucket, video);

if (GetSQSMessageSuccess()==true)
 GetPersonDetectionResults();

```

## Java V2

Este código se toma desde el AWS Documentación Ejemplos del SDK de repositorio de GitHub. Ver el ejemplo completo [aquí](#).

```
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

```
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()
```

#### 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 \(SDK\) \(p. 72\)](#), el código que se va a reemplazar podría ser diferente.

- 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 ([p. 803](#)) que contienen detalles acerca de las personas detectadas en el video 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 video. Especifique INDEX para ordenar por personas de las que se hace seguimiento en el video. 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 video, durante el que se realiza un seguimiento de los recorridos de las personas en el video. En la respuesta, tenga en cuenta lo siguiente:

- Información de persona— El PersonDetectionEl elemento de matriz contiene información acerca de la persona detectada. Por ejemplo, el momento en que se detectó a la persona (Timestamp), la posición de la persona en un fotograma de vídeo a la hora en la que se detectan (BoundingBox) y la confianza de Amazon Rekognition Video de que la persona ha sido detectada 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 \(p. 68\)](#).
- Índice: un identificador único para identificar a 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": 0.2166666865348816
 },
 "Face": {
 "BoundingBox": {
 "Height": 0.20000000298023224,
 "Left": 0.02999999329447746,
 "Top": 0.219999988079071,
 "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 personal

Amazon Rekognition puede detectar el equipo de protección personal (EPP) que usan personas en una imagen. Puede utilizar esta información para mejorar las prácticas de seguridad en el lugar de trabajo. Por ejemplo, puede utilizar la detección de EPP para ayudar a determinar si los trabajadores de una obra usan fundas para la cabeza o si los trabajadores médicos usan fundas faciales y cubremanos. La siguiente imagen muestra algunos de los tipos de EPI que se pueden detectar.



Para detectar EPP en una imagen, llame al [DetectProtectiveEquipment \(p. 592\)](#) API y pase una imagen de entrada. La respuesta es una estructura JSON que incluye lo siguiente.

- Las personas detectadas en la imagen.
- Partes de un cuerpo donde se usa el EPP (cara, cabeza, mano izquierda y mano derecha).
- Tipos de EPP detectados en las partes de la carrocería (cubierta facial, cubierta de mano y cubierta de la cabeza).
- Para los elementos de EPP detectados, un indicador de si el EPI cubre o no la parte del cuerpo correspondiente.

Se devuelven cuadros delimitadores para las ubicaciones de personas y elementos de EPP detectados en la imagen.

Opcionalmente, puede solicitar un resumen de los elementos de EPP y de las personas detectadas en una imagen. Para obtener más información, consulte [Resumen de EPP detectado en una imagen \(p. 252\)](#).

## Note

La detección de EPP de Amazon Rekognition no realiza reconocimiento facial ni comparación facial y no puede identificar a las personas detectadas.

## Tipos de EPP

[DetectProtectiveEquipment](#) (p. 592) detecta los siguientes tipos de EPP. Si desea detectar otros tipos de EPP en las imágenes, considere utilizar etiquetas personalizadas de Amazon Rekognition para entrenar un modelo personalizado. Para obtener más información, consulte [Etiquetas personalizadas de Amazon Rekognition](#).

### Cubierta de rostros

[DetectProtectiveEquipment](#) puede detectar cubiertas faciales comunes como quirúrgicas, N95 y máscaras hechas de tela.

### Funda de mano

[DetectProtectiveEquipment](#) puede detectar fundas para manos como guantes quirúrgicos y guantes de seguridad.

### Tapa de cabeza

[DetectProtectiveEquipment](#) puede detectar sombreros y cascos duros.

La API indica que se ha detectado una cubierta de cabeza, mano o cara en una imagen. La API no devuelve información sobre el tipo de una portada específica. Por ejemplo, «guante quirúrgico» para el tipo de funda de mano.

## Fianza de detección de EPP

Amazon Rekognition hace una predicción sobre la presencia de EPP, personas y partes del cuerpo en una imagen. La API proporciona una puntuación (50-100) que indica la confianza de Amazon Rekognition en la precisión de una predicción.

#### Note

Si planea utilizar el [DetectProtectiveEquipment](#) para tomar una decisión que afecte a los derechos, la privacidad o el acceso a los servicios de una persona, le recomendamos que transmita el resultado a un humano para su revisión y validación antes de tomar medidas.

## Resumen de EPP detectado en una imagen

Si lo desea, puede solicitar un resumen de los elementos de EPP y de las personas detectadas en una imagen. Puede especificar una lista de los equipos de protección necesarios (cubierta facial, cubierta de mano o cubierta para la cabeza) y un umbral de confianza mínimo (por ejemplo, 80%). La respuesta incluye un resumen de identificador (ID) consolidado por imagen de personas con el EPP requerido, personas sin el EPP requerido y personas en las que no se pudo determinar.

El resumen le permite responder rápidamente a preguntas como ¿Cuántas personas no llevan fundas faciales? o ¿Todos llevan EPP? Cada persona detectada en el resumen tiene un identificador único. Puede utilizar el ID para obtener información, como la ubicación del cuadro delimitador de una persona que no lleva EPP.

#### Note

El ID se genera aleatoriamente sobre una base de análisis por imagen y no es coherente en las imágenes o en varios análisis de la misma imagen.

Puedes resumir las fundas faciales, las fundas para la cabeza, las fundas de manos o una combinación de tu elección. Para especificar los tipos de EPP necesarios, consulte[Especificación de los requisitos de resumen \(p. 254\)](#). 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 acerca de la respuesta de resumen de[DetectProtectiveEquipment](#), consulte[Descripción de la respuesta de DetectProtective Equipment \(p. 254\)](#).

## Tutorial: Creación de un valorAWS Lambdafunción que detecta imágenes con EPP

Puede crear unAWS LambdaFunción que detecta el equipo de protección personal (EPI) en imágenes ubicadas en un bucket de Amazon S3. Consulte el[AWS Documentación Ejemplos del SDK repositorio de GitHub](#)para este tutorial de Java V2.

## Descripción de la API de detección de equipos de protección personal

La siguiente información describe el[DetectProtectiveEquipment \(p. 592\)](#)API. Para ver código de ejemplo, consulte[Detección de equipos de protección personal en una imagen \(p. 258\)](#).

### Suministro de una imagen

Puede proporcionar la imagen de entrada (formato JPG o PNG) como bytes de imagen o hacer referencia a una imagen almacenada en un bucket de Amazon S3.

Recomendamos utilizar imágenes donde la cara de la persona esté orientada hacia la cámara.

Si la imagen de entrada no gira a una orientación de 0 grados, recomendamos girarla a una orientación de 0 grados antes de enviarla a[detectProtectiveEquipment](#). Las imágenes en formato JPG pueden contener información de orientación en metadatos de formato de archivo de imagen intercambiable (EXIF). Puede utilizar esta información para escribir código que rote la imagen. Para obtener más información, consulte[Exif versión 2.32](#). Las imágenes en formato PNG no contienen información de orientación de imagen.

Para pasar una imagen desde un bucket de Amazon S3, utilice un usuario de IAM con al menosPrivilegios de AmazonS3ReadOnlyAccess. Utilice un usuario de IAM conAmazonRekognitionFullAccessprivilegios para llamar[DetectProtectiveEquipment](#).

En el siguiente ejemplo de entrada de JSON, la imagen se pasa en un bucket de Amazon S3. Para obtener más información, consulte[Trabajar con imágenes \(p. 28\)](#). En el ejemplo se solicita un resumen de todos los tipos de EPP (cubierta de cabeza, cubierta de mano y cubierta facial) con una confianza mínima de detección (`MinConfidence`) del 80%. Debe especificar un`MinConfidence`valor que está entre 50-100% como[detectProtectiveEquipment](#)devuelve predicciones solo 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 [Especificación de los requisitos de resumen \(p. 254\)](#).

```
{
 "Image": {
 "S3Object": {
 "Bucket": "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 llamadas a `DetectProtectiveEquipment` en lotes en segundo plano.

## Especificación de los requisitos de resumen

Opcionalmente, puede utilizar el `SummarizationAttributes` ([the section called “ProtectiveEquipmentSummarizationAttributes” \(p. 813\)](#)) para solicitar información de resumen de los tipos de EPP detectados en una imagen.

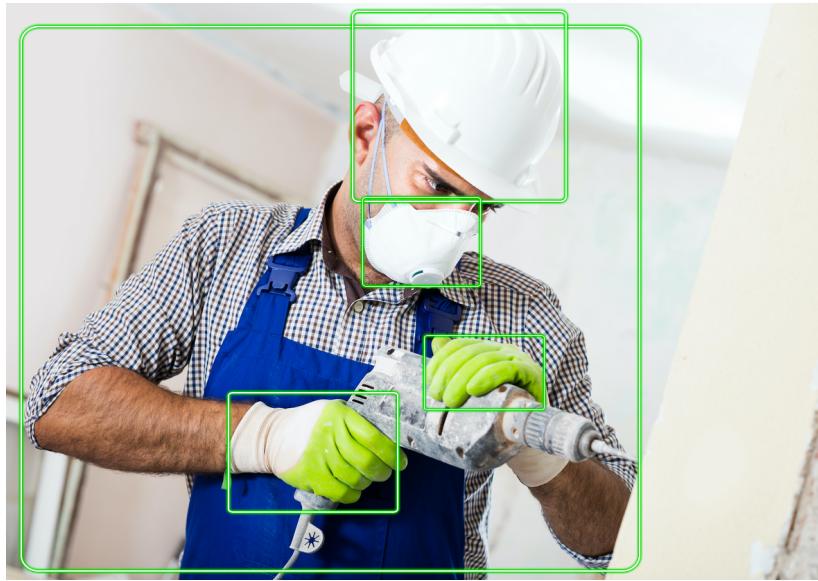
Para especificar los tipos de EPP que se van a resumir, utilice el `RequiredEquipmentTypes` matriz. En la matriz, incluya una o varias de las `FACE_COVER`, `HAND_COVER` o `HEAD_COVER`.

Usar `MinConfidence` para especificar una confianza mínima de detección (50-100). El resumen no incluye las personas, las partes del cuerpo, la cobertura de la parte del cuerpo ni los elementos de EPP, detectados con una confianza inferior a `MinConfidence`.

Para obtener más información acerca de la respuesta de resumen de `DetectProtectiveEquipment`, consulte [Descripción de la respuesta de DetectProtective Equipment \(p. 254\)](#).

## Descripción de la respuesta de DetectProtective Equipment

`DetectProtectiveEquipment` Devuelve un array de personas detectadas en la imagen de entrada. Para cada persona, se devuelve información sobre las partes del cuerpo detectadas y los elementos detectados de EPP. El JSON de la siguiente imagen de un trabajador que lleva una cubierta para la cabeza, una cubierta de mano y una cubierta facial es el siguiente.



En el JSON, observe lo siguiente.

- Personas detectadas—`Persons` es un conjunto de personas detectadas en la imagen (incluidas las personas que no usan EPP). `DetectProtectiveEquipment` puede detectar EPP en un máximo de 15 personas detectadas en una imagen. Cada [ProtectiveEquipmentPerson \(p. 812\)](#) objeto de la matriz contiene un ID de persona, un cuadro delimitador para la persona, partes del cuerpo detectadas y elementos detectados de EPP. El valor `deConfidence` en `ProtectiveEquipmentPerson` indica el porcentaje de confianza de que Amazon Rekognition tiene de que el cuadro delimitador contiene una persona.
- Partes de carrocería—`BodyParts` es un conjunto de partes de carrocería ([ProtectiveEquipmentBodyPart \(p. 811\)](#)) detectado en una persona (incluidas las partes del cuerpo no cubiertas por EPP). Cada `ProtectiveEquipmentBodyPart` incluye el nombre (`Name`) de la parte del cuerpo detectada. `DetectProtectiveEquipment` puede detectar partes del cuerpo de la cara, la cabeza, la izquierda y la mano derecha. La `Confidence` campo en `ProtectiveEquipmentBodyPart` indica el porcentaje de confianza que Amazon Rekognition tiene en la precisión de detección de la parte del cuerpo.
- Elementos de EPI— Matriz `EquipmentDetections` en un `ProtectiveEquipmentBodyPart` objeto contiene una matriz de elementos de PPE detectados. Cada [EquipmentDetection \(p. 767\)](#) contiene los siguientes campos.
  - `Type`— Tipo de EPP detectado.
  - `BoundingBox`: un cuadro delimitador en torno al EPP detectado.
  - `Confidence`: la confianza de Amazon Rekognition de que el cuadro delimitador contiene el EPI detectado.
  - `CoversBodyPart`— Indica si el EPP detectado se encuentra en la parte del cuerpo correspondiente.

La sección titulada “[CoversBodyPart \(p. 752\)](#)” campo `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. Puede usar `CoversBodyPart` para filtrar los casos en que el EPP detectado se encuentra en la imagen, pero no en realidad en la persona.

#### Note

`CoversBodyPart` no indica, ni implica, que la persona esté debidamente protegida por el equipo de protección ni que el propio equipo de protección esté debidamente usado.

- Información resumida—`Summary` contiene la información resumida especificada en el `SummarizationAttributes` parámetro de entrada. Para obtener más información, consulte [Especificación de los requisitos de resumen \(p. 254\)](#).

`Summary` es un objeto de tipo [the section called “ProtectiveEquipmentSummary” \(p. 814\)](#) que contiene la siguiente información:

- `PersonsWithRequiredEquipment`— Una serie de identificaciones de personas en las que cada persona cumple los siguientes criterios.
  - La persona lleva todo el EPP especificado en el `SummarizationAttributes` parámetro de entrada.
  - La `Confidence` para la persona (`ProtectiveEquipmentPerson`), parte del cuerpo (`ProtectiveEquipmentBodyPart`), equipo de protección (`EquipmentDetection`) es igual o mayor que el umbral mínimo de confianza especificado (`MinConfidence`).
  - El valor de `CoversBodyPart` para todos los elementos de EPP es cierto.
- `PersonsWithoutRequiredEquipment`: una matriz de los identificadores de personas que cumplen alguno de los siguientes criterios.
  - La `Confidence` valor para la persona (`ProtectiveEquipmentPerson`), parte del cuerpo (`ProtectiveEquipmentBodyPart`) y cobertura de parte del cuerpo (`CoversBodyPart`) son superiores al umbral de confianza mínimo especificado (`MinConfidence`), pero a la persona le falta uno o varios EPP especificados (`SummarizationAttributes`).
  - El valor de `CoversBodyPart` falso para cualquier EPP especificado (`SummarizationAttributes`) que tiene `unConfidence` valor superior al umbral de confianza mínimo especificado (`MinConfidence`). La persona también tiene todo el EPP especificado (`SummarizationAttributes`) y el `Confidence` valores para persona (`ProtectiveEquipmentPerson`), parte del cuerpo (`ProtectiveEquipmentBodyPart`) y equipos de protección (`EquipmentDetection`) son mayores o iguales que el umbral mínimo de confianza (`MinConfidence`).
- `PersonsIndeterminate`— Una matriz de los identificadores de las personas detectadas cuando el `Confidence` valor para la persona (`ProtectiveEquipmentPerson`), parte del cuerpo (`ProtectiveEquipmentBodyPart`), equipo de protección (`EquipmentDetection`), o `CoversBodyPart` booleano es inferior al umbral de confianza mínimo especificado (`MinConfidence`).

Utilice el tamaño de matriz para obtener un recuento de un resumen concreto. Por ejemplo, el tamaño de `PersonsWithRequiredEquipment` indica el número de personas detectadas que llevan el tipo de EPP especificado.

Puede utilizar el ID de persona para obtener más información sobre una persona, como la ubicación del cuadro delimitador de la persona. El ID de persona se asigna al campo ID de `unProtectiveEquipmentPerson` objeto devuelto en `Persons`(matriz `deProtectiveEquipmentPerson`). A continuación, puede obtener el cuadro Delimitador y otra información del correspondiente `ProtectiveEquipmentPerson` objeto.

```
{
 "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": 0.94122254848022,
 "Left": 0.02214839495718479,
 "Top": 0.03134796395897865
},
"Confidence": 99.98855590820312,
"Id": 0
},
"Summary": {
 "PersonsWithRequiredEquipment": [
 0
],
 "PersonsWithoutRequiredEquipment": [],
 "PersonsIndeterminate": []
}
}
```

## Detección de equipos de protección personal en una imagen

Para detectar equipos de protección personal (EPP) en personas que aparecen en una imagen, utilice el [DetectProtectiveEquipment \(p. 592\)](#) Operación de API sin 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 el 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 \(p. 28\)](#).

Para detectar EPP en personas de una imagen

1. Si aún no lo ha hecho:
  - a. Crear o actualizar un usuario de IAM con `AmazonRekognitionFullAccess` y `AmazonS3ReadOnlyAccess` permisos. Para obtener más información, consulte [Paso 1: Configurar una cuenta de AWS y crear un usuario de IAM \(p. 13\)](#).
  - b. Instale y configure la AWS CLI y los AWS SDK. Para obtener más información, consulte [Paso 2: Configurar la AWS CLI y el AWS SDK de \(p. 14\)](#).
2. Cargue una imagen (que contenga una o varias personas que usan EPP) en su bucket de S3.  
Para obtener instrucciones, consulte [Carga de objetos en Amazon S3](#) en la Amazon Simple Storage Service User Guide.
3. Utilice los siguientes ejemplos para llamar a la operación `DetectProtectiveEquipment`. Para obtener más información acerca de la visualización de cuadros delimitadores en una imagen, consulte [Visualización de cuadros delimitadores \(p. 49\)](#).

## Java

En este ejemplo se muestra información acerca de los elementos de EPI detectados en las personas detectadas en una imagen.

Cambie el valor de `bucket` con el nombre del bucket de Amazon S3 que contiene la imagen.  
Cambie el valor de `photo` al nombre de 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 toma de la AWS Documentación de ejemplos de SDK de repositorio de GitHub. Ver el ejemplo completo [aquí](#).

```
public static void displayGear(S3Client s3,
 RekognitionClient rekClient,
 String sourceImage,
 String bucketName) {

 byte[] data = getObjectType(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("No body parts found for ID: " + person.id());
 }
 else {
 System.out.println("Body parts for ID: " + person.id());
 for (ProtectiveEquipmentBodyPart part: bodyParts) {
 System.out.println(part.type());
 }
 }
 }
 } catch (Exception e) {
 e.printStackTrace();
 }
}
```

```
 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);
} 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);
```

```
byte[] data = objectBytes.asByteArray();
return data;

} 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();
}
```

## AWS CLI

Este AWS CLI solicita un resumen de PPE y muestra la salida de JSON para la `detect-protective-equipment` de la CLI.

Cambie `bucketname` con el nombre de un bucket de Amazon S3 que contiene una imagen.  
Cambio `input.jpg` al nombre de la imagen que desea usar.

```
aws rekognition detect-protective-equipment \
--image "S3Object={Bucket=bucketname,Name=input.jpg}" \
--summarization-attributes
"MinConfidence=80,RequiredEquipmentTypes=['FACE_COVER', 'HAND_COVER', 'HEAD_COVER ']"
```

Este comando de la AWS CLI muestra la salida de JSON para la operación `detect-protective-equipment` de la CLI.

Cambie `bucketname` con el nombre de un bucket de Amazon S3 que contiene una imagen.  
Cambio `input.jpg` al nombre de la imagen que desea usar.

```
aws rekognition detect-protective-equipment \
--image "S3Object={Bucket=bucketname,Name=input.jpg}"
```

## Python

En este ejemplo se muestra información acerca de los elementos de EPI detectados en las personas detectadas en una imagen.

Cambie el valor de `bucket` con el nombre del bucket de Amazon S3 que contiene la imagen.  
Cambie el valor de `photo` al nombre de 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.)

import boto3

def detect_labels(photo, bucket):

 client=boto3.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\t' + ppe_item['Type'] + '\n\t\t\t\tConfidence: ' +
str(ppe_item['Confidence']))
 print('\t\t\tCovers body part: ' +
str(ppe_item['CoversBodyPart']['Value']) + '\n\t\t\t\tConfidence: ' +
str(ppe_item['CoversBodyPart']['Confidence']))
 print('\t\t\tBounding Box:')
 print ('\t\t\t\tTop: ' + str(ppe_item['BoundingBox']['Top']))
 print ('\t\t\t\tLeft: ' + str(ppe_item['BoundingBox'])
['Left']))
 print ('\t\t\t\tWidth: ' + str(ppe_item['BoundingBox']
['Width']))
 print ('\t\t\t\tHeight: ' + str(ppe_item['BoundingBox']
['Height']))
 print ('\t\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'
 bucket='bucket'
 person_count=detect_labels(photo, bucket)
 print("Persons detected: " + str(person_count))

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

## Ejemplo: Dibujo de cuadros delimitadores alrededor de cubiertas de caras

En los siguientes ejemplos se muestra cómo dibujar cuadros delimitadores alrededor de las cubiertas de caras detectadas en personas. Para un ejemplo que utiliza AWS Lambda Amazon DynamoDB, consulte la [AWS Documentación Ejemplos del SDK repositorio de GitHub](#).

Para detectar cubiertas faciales, utiliza el [DetectProtectiveEquipment \(p. 592\)](#) Operación de API sin almacenamiento. La imagen se carga desde el sistema de archivos local. Proporciona la imagen de entrada a `detectProtectiveEquipment` como matriz de bytes de imagen (bytes de imagen con codificación base64). Para obtener más información, consulte [Trabajar con imágenes \(p. 28\)](#).

En el ejemplo se muestra un cuadro delimitador alrededor de las cubiertas de caras detectadas. El cuadro delimitador es verde si la cubierta frontal cubre completamente la parte del cuerpo. De lo contrario, se mostrará un cuadro delimitador rojo. Como advertencia, se muestra un cuadro delimitador amarillo dentro del cuadro delimitador de la cubierta frontal, si la confianza de detección es inferior al valor de confianza especificado. Si no se detecta una cubierta facial, se dibuja un cuadro delimitador rojo alrededor de la persona.

La salida de la imagen es similar a la siguiente.



Para mostrar cuadros delimitadores en las cubiertas de caras detectadas

1. Si aún no lo ha hecho:
  - a. Crear o actualizar un usuario de IAM con `AmazonRekognitionFullAccess` permisos. Para obtener más información, consulte [Paso 1: Configurar una cuenta de AWS y crear un usuario de IAM \(p. 13\)](#).
  - b. Instale y configure la AWS CLI y los AWS SDK. Para obtener más información, consulte [Paso 2: Configurar la AWS CLI y AWS SDK de \(p. 14\)](#).
2. Utilice los siguientes ejemplos para llamar a la operación `DetectProtectiveEquipment`. Para obtener más información acerca de la visualización de cuadros delimitadores en una imagen, consulte [Visualización de cuadros delimitadores \(p. 49\)](#).

Java

En la función `main`, cambie lo siguiente:

- El valor de `photoruta` y 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.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 toma de la AWS Documentación de ejemplos de SDK de repositorio de GitHub. Ver el ejemplo completo [aquí](#).

```
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);

 // Create frame and panel.
 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 (IOException e) {
 e.printStackTrace();
 } 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);
byte[] data = objectBytes.asByteArray();
return data;

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

public PPEBoundingBoxFrame(DetectProtectiveEquipmentResponse ppeResult,
BufferedImage bufImage, float requiredConfidence) throws Exception {
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) {
 BoundingBox boxPerson = person.boundingBox();
 left = width * boxPerson.left();
 top = height * boxPerson.top();
 Boolean foundMask=false;

 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)
 {
```

Python

En la función `main`, cambie lo siguiente:

- El valor de `photoruta` y nombre de archivo de un archivo de imagen local (PNG o JPEG).
  - El valor de `confidence` al nivel de confianza deseado (50-100).

```
#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()
```

# Reconocimiento de famosos

Amazon Rekognition Amazon Rekognition facilita a los clientes reconocer automáticamente decenas de miles de personalidades conocidas en imágenes y vídeos mediante el aprendizaje automático. Los metadatos proporcionados por la API de reconocimiento de celebridades reducen significativamente el esfuerzo manual repetitivo necesario para etiquetar contenido y hacer que se pueda buscar fácilmente.

La rápida proliferación de contenido de imagen y vídeo significa que las empresas de medios a menudo tienen dificultades para organizar, buscar y utilizar sus catálogos multimedia a escala. Los canales de noticias y las emisoras deportivas a menudo necesitan encontrar imágenes y vídeos rápidamente para responder a los eventos actuales y crear una programación relevante. Los metadatos insuficientes dificultan estas tareas, pero con Amazon Rekognition puedes etiquetar automáticamente grandes volúmenes de contenido nuevo o de archivo para que sea fácil buscar un conjunto completo de celebridades internacionales y ampliamente conocidas, como actores, deportistas y creadores de contenido online.

El reconocimiento de famosos de Amazon Rekognition está diseñado para ser utilizados exclusivamente en aquellos casos en los que prevea que es posible que haya un famoso conocido 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 \(p. 181\)](#).

## Note

Si eres una persona famosa y no deseas que se te incluya en esta característica, ponte en contacto con [AWSSoporte](#) o correo electrónico <[rekognition-celebrity-opt-out@amazon.com](mailto:rekognition-celebrity-opt-out@amazon.com)>.

## Temas

- [Reconocimiento de celebridades en comparación con la búsqueda facial \(p. 275\)](#)
- [Reconocimiento de famosos en una imagen \(p. 276\)](#)
- [Reconocimiento de famosos en un vídeo almacenado \(p. 284\)](#)
- [Obtención de información sobre un famoso \(p. 294\)](#)

## Reconocimiento de celebridades en comparación con la búsqueda facial

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 capacitado para reconocer a cientos de miles de personas populares 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 coincidir rostros entre diferentes 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.

En cambio, 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 \(p. 181\)](#).

## 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 \(p. 671\)](#). Por ejemplo, en las redes sociales o noticias y en los sectores de noticias y entretenimiento, donde el factor temporal de recopilación de la información puede ser crítico, puede utilizar el `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 \(p. 602\)](#). 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 se añade una nueva imagen a su colección, puede utilizar una función AWS Lambda para reconocer a famosos llamando a `RecognizeCelebrities` cuando la imagen se carga en un bucket de S3.

## Llamar a `RecognizeCelebrities`

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.AWS Command Line Interface(AWS CLI) o el SDK de AWS. En el procedimiento de AWS CLI, carga una imagen en formato .jpg o .png en un bucket de S3. En los procedimientos del SDK de AWS, utiliza una imagen cargada desde su sistema de archivos local. Para obtener información sobre recomendaciones de imagen de entrada, consulte [Trabajar con imágenes \(p. 28\)](#).

Para ejecutar este procedimiento, necesitará un archivo de imagen que contenga uno o varios rostros de famosos.

Para reconocer famosos en una imagen

1. Si aún no lo ha hecho:
  - a. Crear o actualizar un usuario de IAM con `AmazonRekognitionFullAccess` y `AmazonS3ReadOnlyAccess` permisos. Para obtener más información, consulte [Paso 1: Configurar una cuenta de AWS y crear un usuario de IAM \(p. 13\)](#).

- b. Instale y configure la AWS CLI y los AWS SDK. Para obtener más información, consulte [Paso 2: Configurar la AWS CLI y AWSSDK de \(p. 14\)](#).
2. Utilice 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 toma desde el AWS Documentación Ejemplos del SDK repositorio de GitHub. Ver el ejemplo completo [aquí](#).

```
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);
 }
}
```

```
}
```

## AWS CLI

Este comando de la AWS CLI muestra la salida de JSON para la operación `recognize-celebrities` de la CLI.

Cambie `bucketname` con el nombre del bucket de Amazon S3 que contenga una imagen. Cambie el valor de `input.jpg` por el nombre de un archivo de imagen que contenga uno o más rostros de famosos.

```
aws rekognition recognize-celebrities \
--image "S3Object={Bucket=bucketname,Name=input.jpg}"
```

## 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.

```
#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

def recognize_celebrities(photo):

 client=boto3.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'])
 print ('Smile: ' + celebrity['Smile'])
 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='moviestars.jpg'

 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 debucketPara el nombre del bucket de S3 que contenga el archivo de imagen proporcionado. Cambie el valor deREGIONAl nombre de la región asociada a tu cuenta.

```
// 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"
// Create SNS service object.
const rekogClient = new RekognitionClient({ region: REGION });

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 celebrityUrls)
 Console.WriteLine(url);
 }
 Console.WriteLine(recognizeCelebritiesResponse.UnrecognizedFaces.Count + "
face(s) were unrecognized.");
 }
}
```

- Registre el valor de uno de los ID de famosos que se muestran. Lo necesitará en [Obtención de información sobre un famoso \(p. 294\)](#).

## Solicitud de operación RecognizeCelebrities

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 \(p. 28\)](#).

```
{
 "Image": {
 "Bytes": "/AoSiyyvFpm....."
 }
}
```

## Respuesta de la operación RecognizeCelebrities

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—`CelebrityFaces`es una matriz de famosos reconocidos. Cada [Celebrity \(p. 742\)](#) objeto de la matriz de famosos contiene el nombre del famoso y una lista de las URL que apuntan a contenido relacionado, por ejemplo, el enlace de IMDB o Wikidata del famoso. Amazon Rekognition devuelve un [ComparedFace \(p. 747\)](#) objeto que su aplicación puede utilizar para determinar si el rostro del famoso está en la imagen y un identificador único para el famoso. Utilice el identificador único para recuperar información sobre el famoso más adelante con la operación API [GetCelebrityInfo \(p. 602\)](#).
- Rostros no reconocidos—`UnrecognizedFaces`es una matriz de rostros que no coinciden con ningún famoso conocido. Cada objeto [ComparedFace \(p. 747\)](#) 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": 0.35450000000000003,
 "Y": 0.3333333333333333
 }
 }]
}
```

```
 "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": 9.012002944946289
 }
 }
]
```

```
 "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 famosos en un vídeo almacenado, utilice [StartCelebrityRecognition \(p. 685\)](#) para iniciar análisis de vídeo. Amazon Rekognition Video publica el estado de finalización del análisis de vídeo en un tema de Amazon Simple Notification Service. Si el análisis de vídeo es correcto, llame a [GetCelebrityRecognition \(p. 605\)](#) para obtener los resultados del análisis. Para obtener más información sobre cómo iniciar el análisis de vídeo y obtener los resultados, consulte [Llamar a las operaciones de Amazon Rekognition Video \(p. 66\)](#).

Este procedimiento amplía el código que se describe en [Análisis de un vídeo almacenado en un bucket de Amazon S3 con Java o Python \(SDK\) \(p. 72\)](#), que utiliza una cola de Amazon SQS para obtener el estado de finalización 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 \(SDK\) \(p. 72\)](#).
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(bucket, video);
```

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

por:

```
StartCelebrityDetection(bucket, video);

if (GetSQSMessageSuccess() == true)
 GetCelebrityDetectionResults();
```

## Java V2

Este código se toma desde el AWS Documentación Ejemplos del SDK repositorio de GitHub. Ver el ejemplo completo [aquí](#).

```
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: " + Long.toString(seconds) + " ");
 CelebrityDetail details=celeb.celebrity();
 System.out.println("Name: " + details.name());
 System.out.println("Id: " + details.id());
 System.out.println();
}

} while (recognitionResponse !=null &&
recognitionResponse.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.)

===== 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 Node.Js, sustituya el valor debucketcon el nombre del bucket de S3 que contenga su vídeo y el valor devideoNamecon el nombre del archivo de vídeo. También necesitará reemplazar el valor deroleArncon el Arn asociado a su rol de servicio de IAM. Por último, sustuya el valor dendregioncon el nombre de la región operativa asociada a tu cuenta.

```
// 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";

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

// Set bucket and video variables
const bucket = "bucket-name";
const videoName = "video-name";
const roleArn = "RoleArn"
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",
 Statement: [
 {
 Sid: "MyPolicy",
 Effect: "Allow",
 Principal: {AWS: "*"},
 Action: "SQS:SendMessage",
 Resource: queueArn,
 Condition: {
 ArnEquals: {
 'aws:SourceArn': topicArn
 }
 }
 }
]
 }
 const updatedQueueAttribs = await sqsClient.send(new
UpdateQueueAttributesCommand({QueueUrl: sqsQueueUrl, Attributes: [policy]}))
 console.log(updatedQueueAttribs)
 }
}
```

```
 }
 }
};

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()
 })
 // Search 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 "SUCCEEDED", 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()
```

#### 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 \(SDK\) \(p. 72\)](#), el código que se va a reemplazar podría ser diferente.

- Ejecute el código. Se muestra información sobre los famosos reconocidos en el vídeo.

## Respuesta de operación GetCelebrityRecognition

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 \(p. 746\)](#) por cada momento en que se reconoce al famoso en el vídeo. Cada CelebrityRecognition contiene información sobre un famoso reconocido ([CelebrityDetail \(p. 744\)](#)) 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 acerca de un famoso reconocido. Incluye el nombre de la celebridad ([Name](#)), identificador ([ID](#)), el género conocido de la celebridad ([KnownGender](#)) y una lista de direcciones URL que apuntan a contenido relacionado ([Urls](#)). También incluye el nivel de confianza que Amazon Rekognition Video tiene en la precisión del reconocimiento y detalles acerca del rostro del famoso, [FaceDetail \(p. 773\)](#). Si necesita obtener contenido relacionado más adelante, puede utilizar [ID](#) con [GetCelebrityInfo \(p. 602\)](#).
- VideoMetadata: información sobre el vídeo que se ha analizado.

```
{
 "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 \(p. 602\)](#). El famoso se identifica mediante el ID de famoso que devuelve una llamada a anterior a [the section called “RecognizeCelebrities” \(p. 671\)](#).

### Llamar a 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 \(p. 276\)](#).

Para obtener información sobre un famoso (SDK)

1. Si aún no lo ha hecho:
  - a. Crear o actualizar un usuario de IAM con `AmazonRekognitionFullAccess` y `AmazonS3ReadOnlyAccess` permisos. Para obtener más información, consulte [Paso 1: Configurar una cuenta de AWS y crear un usuario de IAM \(p. 13\)](#).
  - b. Instale y configure la AWS CLI y los AWS SDK. Para obtener más información, consulte [Paso 2: Configurar la AWS CLI y AWS SDK de \(p. 14\)](#).
2. Utilice los siguientes ejemplos para llamar a la operación `GetCelebrityInfo`.

Java

Este ejemplo muestra el nombre y la información sobre un famoso.

Reemplace `id` por uno de los ID de famoso mostrados en [Reconocimiento de famosos en una imagen \(p. 276\)](#).

```
//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 toma desde el AWS Documentación Ejemplos del SDK repositorio de GitHub. Ver el ejemplo completo [aquí](#).

```
public static void getCelebrityInfo(RekognitionClient rekClient, String id) {

 try {
 GetCelebrityInfoRequest info = GetCelebrityInfoRequest.builder()
 .id(id)
 .build();

 GetCelebrityInfoResponse response = rekClient.getCelebrityInfo(info);

 // Display celebrity information.
 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 comando de la AWS CLI muestra la salida de JSON para la operación `get-celebrity-info` de la CLI. Reemplace `ID` por uno de los ID de famoso mostrados en [Reconocimiento de famosos en una imagen \(p. 276\)](#).

```
aws rekognition get-celebrity-info --id ID
```

## Python

Este ejemplo muestra el nombre y la información sobre un famoso.

Reemplace `id` por uno de los ID de famoso mostrados en [Reconocimiento de famosos en una imagen \(p. 276\)](#).

```
#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):
```

```
client=boto3.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="nnnnnnnn"
 celebrity_info=get_celebrity_info(id)

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

#### .NET

Este ejemplo muestra el nombre y la información sobre un famoso.

Reemplace `id` por uno de los ID de famoso mostrados en [Reconocimiento de famosos en una imagen \(p. 276\)](#).

```
//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 celebrityInfoResponseUrls)
 Console.WriteLine(url);
 }
}
```

```
}
```

## Solicitud de operación GetCelebrityInfo

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": "nnnnnnnn"
}
```

## Respuesta de la operación GetCelebrityInfo

GetCelebrityInfo devuelve una matriz (`Urls`) de enlaces a la información sobre el famoso solicitado.

```
{
 "Name": "Celebrity Name",
 "Urls": [
 "www.imdb.com/name/nmnnnnnnn"
]
}
```

# Moderación de contenido

Puede utilizar Amazon Rekognition para detectar contenido inapropiado, no deseado u ofensivo. Puede utilizar las API de moderación de Rekognition en las redes sociales, los medios de difusión, la publicidad y las situaciones de comercio electrónico para crear una experiencia de usuario más segura, proporcionar garantías de seguridad de marca a los anunciantes y cumplir con las regulaciones locales y globales.

Hoy en día, muchas empresas confían totalmente en los moderadores humanos para revisar contenido de terceros o generado por los usuarios, mientras que otras simplemente reaccionan a las quejas de los usuarios para eliminar imágenes, anuncios o videos ofensivos o inapropiados. Sin embargo, los moderadores humanos por sí solos no pueden escalar para satisfacer estas necesidades a la suficiente calidad o velocidad, lo que lleva a una mala experiencia de usuario, altos costos para alcanzar la escala o incluso perder la reputación de la marca. Al utilizar Rekognition para la moderación de imágenes y videos, los moderadores humanos pueden revisar un conjunto de contenido mucho más pequeño, normalmente del 1-5% del volumen total, ya marcado por el aprendizaje automático. Esto les permite centrarse en actividades más valiosas y seguir logrando una cobertura de moderación integral a una fracción del costo existente. Para configurar personal humano y realizar tareas de revisión humana, puede utilizar la Augmented AI de Amazon, que ya está integrada con Rekognition.

## Temas

- [Uso de las API de moderación de imagen y vídeo \(p. 298\)](#)
- [Detección de imágenes inapropiadas \(p. 300\)](#)
- [Detección de videos almacenados inapropiados \(p. 305\)](#)
- [Revisión de contenido inapropiado con Amazon Augmented AI \(p. 311\)](#)

## Uso de las API de moderación de imagen y vídeo

En la API de imagen de Amazon Rekognition, puede utilizar el [DetectModerationLabels \(p. 588\)](#) para detectar contenido inapropiado u ofensivo en imágenes. Puede utilizar la API de Amazon Rekognition Video para detectar contenido inapropiado de forma asíncrona utilizando el [StartContentModeration \(p. 689\)](#) y [GetContentModeration \(p. 611\)](#) operaciones.

Amazon Rekognition utiliza una taxonomía jerárquica de dos niveles para etiquetar categorías de contenido inapropiado u ofensivo. Cada categoría de nivel superior tiene una serie de categorías de segundo nivel.

| Categoría de nivel superior | Categoría de segundo nivel             |
|-----------------------------|----------------------------------------|
| Desnudo explícito           | Desnudo                                |
|                             | Desnudo masculina gráfica              |
|                             | Desnudo femenino gráfico               |
|                             | Actividad sexual                       |
|                             | Desnudo explícito ilustrado            |
|                             | Juguetes para adultos                  |
| Insinuante                  | Ropa de baño o ropa interior femenina  |
|                             | Ropa de baño o ropa interior masculina |
|                             | Desnudo parcial                        |

| Categoría de nivel superior | Categoría de segundo nivel |
|-----------------------------|----------------------------|
|                             | Hombre con barechested     |
|                             | Ropa provocativa           |
|                             | Situaciones sexuales       |
| Violencia                   | Violencia gráfica o sangre |
|                             | Violencia física           |
|                             | Violencia con armas        |
|                             | Armas                      |
|                             | Autolesiones               |
| Visualmente perturbador     | Cuerpos desnutridos        |
|                             | Cadáveres                  |
|                             | Personas colgadas          |
|                             | Air Crash                  |
|                             | Explosiones y explosiones  |
| Gestos groseros             | Dedo medio                 |
| Drogas                      | Productos de medicamentos  |
|                             | Consumo de drogas          |
|                             | Pastillas                  |
|                             | Parafernalia farmacológica |
| Tabaco                      | Productos de tabaco        |
|                             | Fumar                      |
| Alcohol                     | Beber                      |
|                             | Bebidas alcohólicas        |
| Juegos de azar              | Juegos de azar             |
| Símbolos de                 | Partido nazi               |
|                             | Supremacía Blanca          |
|                             | Extremista                 |

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](#) (p. 796) devuelta por `DetectModerationLabels` (imágenes) y por `GetContentModeration` (vídeos).

Puede configurar el umbral de confianza que Amazon Rekognition utiliza para detectar contenido inadecuado especificando el `MinConfidence` parámetro de entrada. No se devuelven etiquetas para el contenido inadecuado detectado con una confianza menor a `MinConfidence`.

Si se especifica un valor para `MinConfidence` que sea inferior a un 50%, es probable que se devuelva un número elevado de falsos positivos. Le recomendamos que utilice un valor inferior al 50 % solo cuando sea aceptable que la detección tenga una precisión menor. Si no especifica un valor para `MinConfidence`, Amazon Rekognition devuelve etiquetas para el contenido no apropiado detectado con al menos un 50% de confianza.

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 podría devolver «Desnudo explícito» con una alta puntuación de confianza 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, consulte [Detección de imágenes inapropiadas \(p. 300\)](#).

Amazon Rekognition Image y Amazon Rekognition Video devuelven la versión del modelo de detección de moderación que se utiliza para detectar contenido inadecuado (`ModerationModelVersion`).

**Note**

Amazon Rekognition no es una autoridad en la materia ni pretende en modo alguno ser un filtro exhaustivo de contenido inapropiado u ofensivo. Además, las API de moderación de imágenes y vídeo no detectan si una imagen incluye contenido ilegal, como pornografía infantil.

## Detección de imágenes inapropiadas

Puede utilizar el [DetectModerationLabels \(p. 588\)](#) para determinar si una imagen incluye contenido inapropiado u ofensivo. Para obtener una lista de etiquetas de moderación en Amazon Rekognition, consulte [Uso de las API de moderación de imagen y vídeo](#).

## Detección de contenido inadecuado 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 AWS CLI o el SDK de AWS adecuado. Para obtener más información, consulte [Introducción a Amazon Rekognition \(p. 12\)](#). 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. Crear o actualizar un usuario de IAM con `AmazonRekognitionFullAccess` y `AmazonS3ReadOnlyAccess` permisos. Para obtener más información, consulte [Paso 1: Configurar una cuenta de AWS y crear un usuario de IAM \(p. 13\)](#).
  - b. Instale y configure la AWS CLI y los AWS SDK. Para obtener más información, consulte [Paso 2: Configurar la AWS CLI y AWSSDK de \(p. 14\)](#).
2. Cargue una imagen en su bucket de S3.

Para obtener instrucciones, consulte [Carga de objetos en Amazon S3](#) en la Amazon Simple Storage Service.
3. Utilice los siguientes ejemplos para llamar a la operación `DetectModerationLabels`.

## Java

El resultado de este ejemplo son los nombres de etiqueta de contenido no apropiados, los niveles de confianza y la etiqueta principal de las etiquetas de moderación detectadas.

Reemplace los valores de 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 toma de la AWS Ejemplos de SDK de documentación repositorio de GitHub. Ver el ejemplo completo[aquí](#).

```
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);

 // Display the results
 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);
 }
}
```

## AWS CLI

Este comando de la AWS CLI muestra la salida de JSON para la operación `detect-moderation-labels` de la CLI.

Reemplace `bucket` y `input.jpg` por el nombre del bucket de S3 y el nombre de archivo de imagen que utilizó en el Paso 2.

```
aws rekognition detect-moderation-labels \
--image '{"S3Object":{"Bucket":"'bucket'", "Name":"'input.jpg"}'}
```

## Python

El resultado de este ejemplo son los nombres de las etiquetas de contenido no apropiado u ofensivo detectadas, los niveles de confianza y la etiqueta principal de las etiquetas de contenido no apropiado detectadas.

En la función `main`, reemplace los valores de `bucket` y `photo` por el nombre del bucket de S3 y el nombre del 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.)

import boto3

def moderate_image(photo, bucket):

 client=boto3.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='photo'
 bucket='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, los niveles de confianza y la etiqueta principal detectadas, así como la etiqueta principal de las etiquetas de moderación detectadas.

Reemplace los valores de `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 = "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);
}
}
}
}
```

## Solicitud de operación DetectModerationLabels

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 Amazon Rekognition Image debe tener en la precisión de la etiqueta detectada para que se devuelva en la respuesta.

```
{
 "Image": {
 "S3Object": {
 "Bucket": "bucket",
 "Name": "input.jpg"
 }
 },
 "MinConfidence": 60
}
```

## Respuesta de la operación DetectModerationLabels

`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 imagen inadecuada: el ejemplo muestra una lista de etiquetas para el contenido inapropiado u ofensivo encontrado 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 principal de una etiqueta de nivel superior es "".

Fianza 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": 99.24723052978516,
 "ParentName": "",
 "Name": "Explicit Nudity"
 },
 {
 "Confidence": 99.24723052978516,
 "ParentName": "Explicit Nudity",
 "Name": "Graphic Male Nudity"
 },
 {
 "Confidence": 88.25341796875,
 "ParentName": "Explicit Nudity",
 "Name": "Sexual Activity"
 }
]
}
```

## Detección de vídeos almacenados inapropiados

Amazon Rekognition Video La detección de contenido inapropiado u ofensivo en vídeos almacenados es una operación asíncrona. Para empezar a detectar contenido inapropiado u ofensivo, llame [StartContentModeration \(p. 689\)](#). Amazon Rekognition Video publica el estado de la realización del análisis de vídeo en un tema de Amazon Simple Notification Service. Si el análisis de vídeo es correcto, llame a [GetContentModeration \(p. 611\)](#) para obtener los resultados del análisis. Para obtener más información sobre cómo iniciar el análisis de vídeo y obtener los resultados, consulte [Llamar a las operaciones de Amazon Rekognition Video \(p. 66\)](#). Para obtener una lista de etiquetas de moderación en Amazon Rekognition, consulte [Uso de las API de moderación de imagen y vídeo](#).

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 \(SDK\) \(p. 72\)](#), 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 \(SDK\) \(p. 72\)](#).
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(bucket, video);

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

por:

```
StartUnsafeContentDetection(bucket, video);

if (GetSQSMessageSuccess()==true)
 GetUnsafeContentDetectionResults();
```

## Java V2

Este código se toma de la AWS Ejemplos de SDK de documentación repositorio de GitHub. Ver el ejemplo completo[aquí](#).

```
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)

 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 \(SDK\) \(p. 72\)](#), el código que se va a reemplazar podría ser diferente.

3. Ejecute el código. Se muestra una lista de etiquetas de contenido no apropiadas detectadas en el vídeo.

## Respuesta de la operación GetContentModeration

La respuesta de GetContentModeration es una matriz, `ModerationLabels` de objetos [ContentModerationDetection \(p. 751\)](#). La matriz contiene un elemento para cada vez que se detecta una etiqueta de contenido inadecuada. Dentro de `unContentModerationDetectionObject` del objeto, `ModerationLabel (p. 796)` contiene información para un elemento detectado de contenido inapropiado u ofensivo. `Timestamp` es el tiempo, en milisegundos desde el principio del vídeo, cuando se detectó la etiqueta. Las etiquetas se organizan jerárquicamente de la misma forma que las etiquetas detectadas por un análisis de imagen de contenido inadecuado. Para obtener más información, consulte [Moderación de contenido \(p. 298\)](#).

A continuación se muestra un ejemplo de respuesta de GetContentModeration.

```
{
 "JobStatus": "SUCCEEDED",
 "ModerationLabels": [
 {
 "ModerationLabel": {
 "Confidence": 93.02153015136719,
 "Name": "Male Swimwear Or Underwear",
 "ParentName": "Suggestive"
 },
 "Timestamp": 0
 },
 {
 "ModerationLabel": {
 "Confidence": 93.02153015136719,
 "Name": "Suggestive",
 "ParentName": ""
 },
 "Timestamp": 0
 },
 {
 "ModerationLabel": {
 "Confidence": 98.29075622558594,
 "Name": "Male Swimwear Or Underwear",
 "ParentName": "Suggestive"
 },
 "Timestamp": 1000
 },
 {
 "ModerationLabel": {
 "Confidence": 98.29075622558594,
 "Name": "Suggestive",
 "ParentName": ""
 },
 "Timestamp": 1000
 },
 {
 "ModerationLabel": {
 "Confidence": 97.91191101074219,
 "Name": "Underwear",
 "ParentName": "Male Swimwear Or Underwear"
 },
 "Timestamp": 1000
 }
]
}
```

```
 "Name": "Male Swimwear Or Underwear",
 "ParentName": "Suggestive"
 },
 "Timestamp": 1999
}
],
"NextToken": "w5xfYx64+QvCdGTidVVWtczKHe0JAcUFu2tJ1RgDevHRovJ
+1xej2GUDfTMWrTVn1nwSMHi9",
"VideoMetadata": {
 "Codec": "h264",
 "DurationMillis": 3533,
 "Format": "QuickTime / MOV",
 "FrameHeight": 1080,
 "FrameRate": 30,
 "FrameWidth": 1920
}
}
```

## Revisión de contenido inapropiado con Amazon Augmented AI

La inteligencia artificial de Amazon (Amazon A2I) le permite crear los flujos de trabajo necesarios para la revisión humana de las predicciones del aprendizaje automático.

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 de 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 Amazon RekognitionDetectModerationLabel. 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 demostración integral de cómo utilizar Amazon A2I con Amazon Rekognition, consulte uno de los siguientes tutoriales en elGuía para desarrolladores de Amazon SageMaker.

- [Demostrar: Introducción a Amazon A2I Console](#)
- [Demostrar: Introducción a la API de Amazon A2I](#)

Para empezar a utilizar la API, también puedes ejecutar un bloc de notas Jupyter de ejemplo. Consulte[Uso de una instancia de cuaderno de SageMaker con Amazon A2I Jupyter Notebook](#)para utilizar el bloc de notas[Integración de Amazon Augmented AI \(Amazon A2I\) con Amazon Rekognition \[Example\]](#)en una instancia de bloc de notas de SageMaker.

### Ejecución de DetectModerationLabels con Amazon A2I

#### Note

Cree todos sus recursos de Amazon A2I y de Amazon Rekognition en la misma región de AWS.

1. Complete los requisitos previos que se enumeran en [Introducción a Amazon Augmented AI en la Documentación de SageMaker](#).

No olvide configurar sus permisos de IAM como en la página [Permisos y seguridad en la Augmented AI de Amazon](#) en la Documentación de SageMaker.

2. Siga las instrucciones de [Creación de un flujo de trabajo de revisión humana](#) en la Documentación de SageMaker.

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.

Dentro de su llamada `CreateFlowDefinition`, debe establecer el `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` `ModerationLabelName`, que establece la clave que requiere revisión humana. Además, tiene `confidence`, que establece el rango porcentual para enviar a revisión humana con `LessThan`, `GreaterThan` y `Equals`. `Sampling` tiene `RandomSamplingPercentage`, que establece un porcentaje de los 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
 }
 }
]
 }
]
 }
)
```

```
 "ConfidenceGreaterThan": 95
 }
}
]
}
)
```

`CreateFlowDefinition` devuelve un `FlowDefinitionArn`, que utilizará en el siguiente paso cuando llame a `DetectModerationLabels`.

- Para obtener más información, consulte [CreateFlowDefinition](#) en la Referencia de la API de SageMaker.
3. Establezca el parámetro `HumanLoopConfig` cuando llame a `DetectModerationLabels`, como en [Detcción de imágenes inapropiadas \(p. 300\)](#). Consulte el paso 4 para ver ejemplos de `unDetectModerationLabels` llame a con `HumanLoopConfig` conjunto.
    - 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 utilizar `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 AWS CLI y AWS SDK for Python (Boto3) ejecutar `DetectModerationLabels` con `HumanLoopConfig` conjunto.

#### AWS SDK for Python (Boto3)

En el siguiente ejemplo de solicitud se utiliza el SDK for Python (Boto3). Para obtener más información, consulte [detect\\_moderation\\_labels](#) en la Referencia de la API de 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

En el siguiente ejemplo de solicitud se utiliza la CLI de AWS. Para obtener más información, consulte [detect-moderation-labels](#) en la [AWS CLI Reference de los comandos](#).

```
$ 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"]}'
```

```
$ 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 corresponde a `DetectModerationLabels` con `HumanLoopConfig` habilitado, Amazon Rekognition llama a la operación de la API de SageMaker `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 [DescribeHumanLoop](#) en la documentación de referencia de API de la inteligencia artificial aumentada de Amazon.

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 mediante Amazon CloudWatch Events cuando se complete la revisión. Para ver qué eventos buscar, consulte [Eventos de CloudWatch](#) en la Documentación de SageMaker.

Para obtener más información, consulte [Introducción a Amazon Augmented AI](#) en la Documentación de SageMaker.

# 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 habilitada para voz 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) de imágenes.

Para la detección de texto en vídeos, puede implementar soluciones como:

- Búsqueda de vídeos para clips con palabras clave de texto específicas, como el nombre de un invitado en un gráfico en un programa de noticias.
- Moderar el contenido para cumplir con los estándares de la organización mediante la detección de texto accidental, blasfemia o spam.
- Búsqueda de todas las superposiciones de texto en la línea temporal del vídeo para su posterior procesamiento, como la sustitución por texto en otro idioma para la internacionalización del contenido.
- Búsqueda de ubicaciones de texto, de modo que otros gráficos se puedan alinear en consecuencia.

Para detectar texto en imágenes en formato JPEG o PNG, utilice la[DetectText \(p. 596\)](#). Para detectar texto de forma asíncrona en vídeo, utilice el[the section called "StartTextDetection" \(p. 718\)](#)y[the section called "GetTextDetection" \(p. 640\)](#)operaciones. Las operaciones de detección de texto de imagen y vídeo admiten la mayoría de las fuentes, incluidas las que tienen un estilo muy elevado. 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 un fotograma de imagen o vídeo.

La[DetectText](#) y [GetTextDetection](#) las operaciones detectan palabras y líneas. UNApalabra es uno o varios caracteres de guión que no están separados por espacios.[DetectText](#) puede detectar hasta 100 palabras en una imagen.[GetTextDetection](#) puede detectar hasta 50 palabras por fotograma de vídeo.

Una palabra consta de uno o varios caracteres de guión 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 frase completa (los puntos no indican el final de una línea). Por ejemplo, Amazon Rekognition detecta un número de matrícula de conducir como una línea. Una línea finaliza cuando no existe texto alineado después de ella o cuando existe un hueco grande entre las palabras, en relación con la longitud de las mismas. Según la brecha entre las palabras, Amazon Rekognition podría detectar varias líneas en texto alineado en la misma dirección. Si una frase abarca varias líneas, la operación devuelve varias líneas.

Considere la siguiente imagen:



Los cuadros azules representan información sobre texto detectado y la ubicación del texto devuelto por el `DetectText`. En este ejemplo, Amazon Rekognition detecta «ET'S», «MONDAY», «but», «keep» y «Smiling» y «Smiling», «guardar» y «Smiling». Amazon Rekognition detecta «IT», «MONDAY», «MONDAY» y «Smiling» como líneas. Para ser detectado, el texto debe ser dentro de una orientación de +/- 90° grados respecto al eje horizontal.

Para ver un ejemplo, consulte [Detección de texto en una imagen \(p. 316\)](#).

#### Temas

- [Detección de texto en una imagen \(p. 316\)](#)
- [Detección de texto en un vídeo almacenado \(p. 328\)](#)

## Detección de texto en una imagen

Puede proporcionar una imagen de entrada como matriz de bytes de imagen (bytes de imagen con codificación en base64) o como objeto Amazon S3. En este procedimiento cargará una imagen JPEG o PNG en su bucket de S3 y especificará el nombre de archivo.

### Para detectar texto en una imagen (API)

1. Si aún no lo ha hecho, complete los siguientes requisitos previos:

- a. Crea o actualiza un AWS Identity and Access Management(IAM) usuario con `AmazonRekognitionFullAccess` y `AmazonS3ReadOnlyAccess` permisos. Para obtener más información, consulte [Paso 1: Configurar una cuenta de AWS y crear un usuario de IAM \(p. 13\)](#).
- b. Instale y configure la AWS Command Line Interface y los AWS SDK. Para obtener más información, consulte [Paso 2: Configurar la AWS CLI y AWS SDK de \(p. 14\)](#).

2. Cargue la imagen que contiene texto en su bucket de S3.

Para obtener 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 la operación `DetectText`.

#### Java

El siguiente código de ejemplo muestra líneas y palabras que se han detectado en una imagen.

Sustituir los valores `debucketyphoto` con los nombres del bucket de S3 e 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 toma desde la AWS Documentación Ejemplos SDK repositorio de GitHub. Ver el ejemplo completo [aquí](#).

```
public static void detectTextLabels(RekognitionClient rekClient, String
sourceImage) {

 try {

 InputStream sourceStream = new FileInputStream(sourceImage);
 SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);
```

```
// Create an Image object for the source image
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);
}
}
```

## AWS CLI

Este comando de la AWS CLI muestra la salida de JSON para la operación `detect-text` de la CLI.

Sustituir los valores `deBucketName` con los nombres del bucket de S3 e imagen que utilizó en el paso 2.

```
aws rekognition detect-text \
--image "S3Object={Bucket=bucketname,Name=input.jpg}"
```

## Python

El siguiente código de ejemplo muestra líneas y palabras que se han detectado en una imagen.

Sustituir los valores `deBucketPhoto` con los nombres del S3 bucket de 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.)

import boto3

def detect_text(photo, bucket):

 client=boto3.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='bucket'
 photo='photo'
 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.

Sustituir los valores de `bucket` y `photo` con los nombres del bucket de S3 e 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 = "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.

Sustituir los valores debucketphotocon los nombres del S3bucket de y de la imagen que utilizó en el paso 2. Sustituir el valor dendregioncon la región que se encuentra en sus credenciales de .aws.

```
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)
 })
 }
});
```

## Solicitud de operación DetectText

En el navegador `DetectText`, proporciona 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 se muestra la imagen cargada desde un bucket de Amazon S3.

```
{
 "Image": {
 "S3Object": {
 "Bucket": "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. Por último, el filtro de confianza del texto le permite eliminar resultados que no sean fiables, como texto borroso o difuminado. 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. El valor de `MinConfidence` predeterminado es 0.
- `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 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.

## Respuesta de la operación DetectText

La operación `DetectText` analiza la imagen y devuelve una matriz, `TextDetections`, donde cada elemento ([TextDetection \(p. 837\)](#)) 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 de guión 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 `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 del texto en una imagen

Para determinar dónde está el texto reconocido en una imagen, utilice la información de cuadro delimitador ([Geometry \(p. 781\)](#)) 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 \(p. 740\)](#)
- Un polígono más detallado compuesto por varias coordenadas X e Y en una matriz [Point \(p. 805\)](#)

Las coordenadas del polígono y el cuadro delimitador muestran dónde se encuentra el texto en la imagen de origen. Los valores de coordenadas son una ratio del tamaño de imagen global. Para obtener más información, consulte [BoundingBox \(p. 740\)](#).

La siguiente respuesta JSON de la operación `DetectText` muestra las palabras y las líneas detectadas en la imagen siguiente.



```
{
 "TextDetections": [
 {
 "Confidence": 90.54900360107422,
 "DetectedText": "IT'S",
 "Geometry": {
 "BoundingBox": {
 "Height": 20,
 "Left": 300,
 "Top": 100,
 "Width": 200
 },
 "Point": [{
 "X": 300, "Y": 100 }, {
 "X": 500, "Y": 100 }, {
 "X": 500, "Y": 120 }, {
 "X": 300, "Y": 120 }]
 },
 "Id": 1,
 "ParentId": null
 },
 {
 "Confidence": 90.54900360107422,
 "DetectedText": "MONDAY",
 "Geometry": {
 "BoundingBox": {
 "Height": 20,
 "Left": 300,
 "Top": 120,
 "Width": 200
 },
 "Point": [{
 "X": 300, "Y": 120 }, {
 "X": 500, "Y": 120 }, {
 "X": 500, "Y": 140 }, {
 "X": 300, "Y": 140 }]
 },
 "Id": 2,
 "ParentId": 1
 },
 {
 "Confidence": 90.54900360107422,
 "DetectedText": "but keep",
 "Geometry": {
 "BoundingBox": {
 "Height": 20,
 "Left": 300,
 "Top": 140,
 "Width": 200
 },
 "Point": [{
 "X": 300, "Y": 140 }, {
 "X": 500, "Y": 140 }, {
 "X": 500, "Y": 160 }, {
 "X": 300, "Y": 160 }]
 },
 "Id": 3,
 "ParentId": 1
 },
 {
 "Confidence": 90.54900360107422,
 "DetectedText": "keep",
 "Geometry": {
 "BoundingBox": {
 "Height": 20,
 "Left": 300,
 "Top": 160,
 "Width": 200
 },
 "Point": [{
 "X": 300, "Y": 160 }, {
 "X": 500, "Y": 160 }, {
 "X": 500, "Y": 180 }, {
 "X": 300, "Y": 180 }]
 },
 "Id": 4,
 "ParentId": 3
 },
 {
 "Confidence": 90.54900360107422,
 "DetectedText": "smiling",
 "Geometry": {
 "BoundingBox": {
 "Height": 20,
 "Left": 300,
 "Top": 180,
 "Width": 200
 },
 "Point": [{
 "X": 300, "Y": 180 }, {
 "X": 500, "Y": 180 }, {
 "X": 500, "Y": 200 }, {
 "X": 300, "Y": 200 }]
 },
 "Id": 5,
 "ParentId": 3
 }
]
}
```

```
"Geometry": {
 "BoundingBox": {
 "Height": 0.10317354649305344,
 "Left": 0.6677391529083252,
 "Top": 0.17569075524806976,
 "Width": 0.1511344909667968
 },
 "Polygon": [
 {
 "X": 0.6677391529083252,
 "Y": 0.17569075524806976
 },
 {
 "X": 0.8188736438751221,
 "Y": 0.17574213445186615
 },
 {
 "X": 0.8188582062721252,
 "Y": 0.278915673494339
 },
 {
 "X": 0.6677237153053284,
 "Y": 0.2788642942905426
 }
]
},
"Id": 0,
"Type": "LINE"
},
{
 "Confidence": 59.411651611328125,
 "DetectedText": "I",
 "Geometry": {
 "BoundingBox": {
 "Height": 0.05955825746059418,
 "Left": 0.2763049304485321,
 "Top": 0.394121915102005,
 "Width": 0.026684552431106567
 },
 "Polygon": [
 {
 "X": 0.2763049304485321,
 "Y": 0.394121915102005
 },
 {
 "X": 0.30298948287963867,
 "Y": 0.3932435214519501
 },
 {
 "X": 0.30385109782218933,
 "Y": 0.45280176401138306
 },
 {
 "X": 0.27716654539108276,
 "Y": 0.453680157661438
 }
]
},
"Id": 1,
"Type": "LINE"
},
{
 "Confidence": 92.76634979248047,
 "DetectedText": "MONDAY",
 "Geometry": {
 "BoundingBox": {
```

```
 "Height": 0.11997425556182861,
 "Left": 0.5545867085456848,
 "Top": 0.34920141100883484,
 "Width": 0.39841532707214355
 },
 "Polygon": [
 {
 "X": 0.5545867085456848,
 "Y": 0.34920141100883484
 },
 {
 "X": 0.9530020356178284,
 "Y": 0.3471102714538574
 },
 {
 "X": 0.9532787799835205,
 "Y": 0.46708452701568604
 },
 {
 "X": 0.554863452911377,
 "Y": 0.46917566657066345
 }
]
},
"Id": 2,
"Type": "LINE"
},
{
 "Confidence": 96.7636489868164,
 "DetectedText": "but keep",
 "Geometry": {
 "BoundingBox": {
 "Height": 0.0756164938211441,
 "Left": 0.634815514087677,
 "Top": 0.5181083083152771,
 "Width": 0.20877975225448608
 },
 "Polygon": [
 {
 "X": 0.634815514087677,
 "Y": 0.5181083083152771
 },
 {
 "X": 0.8435952663421631,
 "Y": 0.52589350938797
 },
 {
 "X": 0.8423560857772827,
 "Y": 0.6015099883079529
 },
 {
 "X": 0.6335763335227966,
 "Y": 0.59372478723526
 }
]
 },
 "Id": 3,
 "Type": "LINE"
},
{
 "Confidence": 99.47185516357422,
 "DetectedText": "Smiling",
 "Geometry": {
 "BoundingBox": {
 "Height": 0.2814019024372101,
 "Left": 0.48475268483161926,
```

```
 "Top": 0.6823741793632507,
 "Width": 0.47539761662483215
 },
 "Polygon": [
 {
 "X": 0.48475268483161926,
 "Y": 0.6823741793632507
 },
 {
 "X": 0.9601503014564514,
 "Y": 0.587857186794281
 },
 {
 "X": 0.9847385287284851,
 "Y": 0.8692590594291687
 },
 {
 "X": 0.5093409419059753,
 "Y": 0.9637760519981384
 }
]
},
"Id": 4,
"Type": "LINE"
},
{
 "Confidence": 90.54900360107422,
 "DetectedText": "IT'S",
 "Geometry": {
 "BoundingBox": {
 "Height": 0.10387301445007324,
 "Left": 0.6685508489608765,
 "Top": 0.17597118020057678,
 "Width": 0.14985692501068115
 },
 "Polygon": [
 {
 "X": 0.6677391529083252,
 "Y": 0.17569075524806976
 },
 {
 "X": 0.8188736438751221,
 "Y": 0.17574213445186615
 },
 {
 "X": 0.8188582062721252,
 "Y": 0.278915673494339
 },
 {
 "X": 0.6677237153053284,
 "Y": 0.2788642942905426
 }
]
 },
 "Id": 5,
 "ParentId": 0,
 "Type": "WORD"
},
{
 "Confidence": 92.76634979248047,
 "DetectedText": "MONDAY",
 "Geometry": {
 "BoundingBox": {
 "Height": 0.11929994821548462,
 "Left": 0.5540683269500732,
 "Top": 0.34858056902885437,
```

```
 "Width": 0.3998897075653076
 },
 "Polygon": [
 {
 "X": 0.5545867085456848,
 "Y": 0.34920141100883484
 },
 {
 "X": 0.9530020356178284,
 "Y": 0.3471102714538574
 },
 {
 "X": 0.9532787799835205,
 "Y": 0.46708452701568604
 },
 {
 "X": 0.554863452911377,
 "Y": 0.46917566657066345
 }
]
},
"Id": 7,
"ParentId": 2,
"Type": "WORD"
},
{
 "Confidence": 59.411651611328125,
 "DetectedText": "I",
 "Geometry": {
 "BoundingBox": {
 "Height": 0.05981886386871338,
 "Left": 0.2779299318790436,
 "Top": 0.3935416042804718,
 "Width": 0.02624112367630005
 },
 "Polygon": [
 {
 "X": 0.2763049304485321,
 "Y": 0.394121915102005
 },
 {
 "X": 0.30298948287963867,
 "Y": 0.3932435214519501
 },
 {
 "X": 0.30385109782218933,
 "Y": 0.45280176401138306
 },
 {
 "X": 0.27716654539108276,
 "Y": 0.453680157661438
 }
]
 },
 "Id": 6,
 "ParentId": 1,
 "Type": "WORD"
},
{
 "Confidence": 95.33189392089844,
 "DetectedText": "but",
 "Geometry": {
 "BoundingBox": {
 "Height": 0.06849122047424316,
 "Left": 0.6350157260894775,
 "Top": 0.5214487314224243,
 "Width": 0.02624112367630005
 }
 }
}
```

```
 "Width": 0.08413040637969971
 },
 "Polygon": [
 {
 "X": 0.6347596645355225,
 "Y": 0.5215170383453369
 },
 {
 "X": 0.719483494758606,
 "Y": 0.5212655067443848
 },
 {
 "X": 0.7195737957954407,
 "Y": 0.5904868841171265
 },
 {
 "X": 0.6348499655723572,
 "Y": 0.5907384157180786
 }
]
},
"Id": 8,
"ParentId": 3,
"Type": "WORD"
},
{
 "Confidence": 98.1954116821289,
 "DetectedText": "keep",
 "Geometry": {
 "BoundingBox": {
 "Height": 0.07207882404327393,
 "Left": 0.7295929789543152,
 "Top": 0.5265749096870422,
 "Width": 0.11196041107177734
 },
 "Polygon": [
 {
 "X": 0.7290706038475037,
 "Y": 0.5251666903495789
 },
 {
 "X": 0.842876672744751,
 "Y": 0.5268880724906921
 },
 {
 "X": 0.8423973917961121,
 "Y": 0.5989891886711121
 },
 {
 "X": 0.7285913228988647,
 "Y": 0.5972678065299988
 }
]
 },
 "Id": 9,
 "ParentId": 3,
 "Type": "WORD"
},
{
 "Confidence": 99.47185516357422,
 "DetectedText": "Smiling",
 "Geometry": {
 "BoundingBox": {
 "Height": 0.3739858865737915,
 "Left": 0.48920923471450806,
 "Top": 0.5900818109512329,
```

```
 "Width": 0.5097314119338989
 },
 "Polygon": [
 {
 "X": 0.48475268483161926,
 "Y": 0.6823741793632507
 },
 {
 "X": 0.9601503014564514,
 "Y": 0.587857186794281
 },
 {
 "X": 0.9847385287284851,
 "Y": 0.8692590594291687
 },
 {
 "X": 0.5093409419059753,
 "Y": 0.9637760519981384
 }
],
 "Id": 10,
 "ParentId": 4,
 "Type": "WORD"
}
]
}
```

## Detección de texto en un vídeo almacenado

Amazon Rekognition La detección de texto de Amazon Rekognition Video en vídeos almacenados es una operación asíncrona. Para empezar a detectar texto, llame [the section called "StartTextDetection" \(p. 718\)](#). Amazon Rekognition Video publica el estado de la realización del análisis de vídeo en un tema de Amazon SNS. Si el análisis de vídeo es correcto, llame a [the section called "GetTextDetection" \(p. 640\)](#) para obtener los resultados del análisis. Para obtener más información sobre cómo iniciar el análisis de vídeo y obtener los resultados, consulte [Llamar a las operaciones de Amazon Rekognition Video \(p. 66\)](#).

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 \(SDK\) \(p. 72\)](#). Utiliza una cola de Amazon SQS para obtener el estado de realización de una solicitud de análisis de vídeo.

Para detectar texto en un vídeo almacenado en un bucket de Amazon S3 (SDK)

- Realice los pasos que se indican en [Análisis de un vídeo almacenado en un bucket de Amazon S3 con Java o Python \(SDK\) \(p. 72\)](#).
- 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(bucket, video);

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

por:

```
StartTextDetection(bucket, video);

if (GetSQSMessageSuccess()==true)
 GetTextDetectionResults();
```

## Java V2

Este código se toma desde la AWS Documentación Ejemplos SDK repositorio de GitHub. Ver el ejemplo completo [aquí](#).

```
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);
}
}
```

## 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()
```

#### 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 \(SDK\) \(p. 72\)](#), 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. 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. El valor de `MinConfidence` predeterminado es 80.
- `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 encuentran solo parcialmente dentro de las regiones, la respuesta es indefinida.

## Respuesta GetTextDetection

`GetTextDetection` devuelve una matriz (`TextDetectionResults`) que contiene información sobre el texto detectado en el vídeo. Un elemento de la matriz, [TextDetection \(p. 837\)](#), 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 `TextDetectionResult`El elemento de matriz contiene información acerca del texto detectado ([TextDetection \(p. 837\)](#)) 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 \(p. 68\)](#).

- 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": 0.8337579369544983,
 "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
 }
]
 }
 }
 }
]
}
```

```
 "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+Qg1LV1MjjKNOD0Z0GusSPg7TONLe
+OZ3P",
 "TextModelVersion": "3.0"
}
```

# Detección de segmentos de vídeo en vídeo almacenado

Amazon Rekognition Video proporciona una API que identifica segmentos útiles de vídeo, como fotogramas negros y créditos finales.

Los espectadores están viendo más contenidos que nunca. En concreto, las plataformas de transmisión libre (OTT) y vídeo bajo demanda (VOD) ofrecen una amplia selección de opciones de contenido en cualquier momento, lugar y 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 se encuentran los créditos de apertura y final en un fragmento de contenido
- Elegir los lugares adecuados para insertar anuncios, como en secuencias de fotogramas negros silenciosos
- Dividir 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 de análisis de medios operativos mediante API de detección de segmentos de vídeo completamente administradas y diseñadas específicamente con tecnología de aprendizaje automático. Mediante el uso de las API de segmento de Amazon Rekognition Video, puede analizar fácilmente grandes volúmenes de vídeos y detectar marcadores como fotogramas negros o cambios de tomas. Obtendrá códigos temporales, marcas temporales y números de fotogramas SMPTE (Society of Motion Picture and Television Engineers) de cada detección. No se requiere experiencia en ML.

Amazon Rekognition Video analiza los vídeos almacenados en un bucket de Amazon Simple Storage Service (Amazon S3). Los códigos de tiempo SMPTE que se devuelven son precisos de fotogramas: Amazon Rekognition Video proporciona el número exacto de fotogramas de un segmento de vídeo detectado y gestiona automáticamente varios formatos de frecuencia de fotogramas de vídeo. 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 acerca de los precios, consulte [Precios de Amazon Rekognition](#).

La detección de segmentos de Amazon Rekognition Video admite dos tipos de tareas de segmentación: [Tomas técnicas \(p. 339\)](#) y [Detección de tomas \(p. 340\)](#).

## Temas

- [Tomas técnicas \(p. 339\)](#)
- [Detección de tomas \(p. 340\)](#)

- Acerca de la API de detección de segmentos de Amazon Rekognition Video (p. 341)
- Uso de la API de segmentos de Amazon Rekognition (p. 341)
- Ejemplo: Detección de segmentos en un vídeo almacenado (p. 346)

## Tomas técnicas

UNAtaco técnico identifica fotogramas negros, barras de color, créditos de apertura, créditos finales, logotipos de estudio y contenido principal del programa en un vídeo.

### Fotogramas negros

Los vídeos suelen contener fotogramas negros, vacíos y sin audio que se utilizan como señales para insertar anuncios o a fin de marcar el final de un segmento de programa, como una escena o créditos de apertura. Con Amazon Rekognition Video, puede detectar secuencias de fotogramas negros para automatizar la inserción de anuncios, empaquetar el contenido para vídeo bajo pedido 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 identificar de forma automática los fotogramas exactos en los que comienzan y terminan los títulos de apertura y cierre de una película o un programa de televisión. Con esta información, puede generar «marcadores de atracón» o mensajes de espectadores interactivos, como «Siguiente episodio» o «Saltar introducción», en aplicaciones de vídeo a petición (VOD). También puedes detectar el primer y último fotograma del contenido del programa de un vídeo. Amazon Rekognition Video está capacitado para manejar una amplia variedad de estilos de crédito de apertura y finalización, que van desde créditos móviles simples 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.

### Secuencias

Las pizarras 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 más. Amazon Rekognition Video puede identificar el inicio y el final de las pizarras, lo que facilita el uso de los metadatos de texto o la eliminación de la pizarra al preparar el contenido para su visualización final.

### Logos de Studio

Los logotipos de estudio son secuencias que muestran los logotipos o emblemas del estudio de producción involucrado en la realización del espectáculo. Amazon Rekognition Video puede detectar estas secuencias para que los usuarios puedan revisarlas para identificar estudios.

## Contenidos

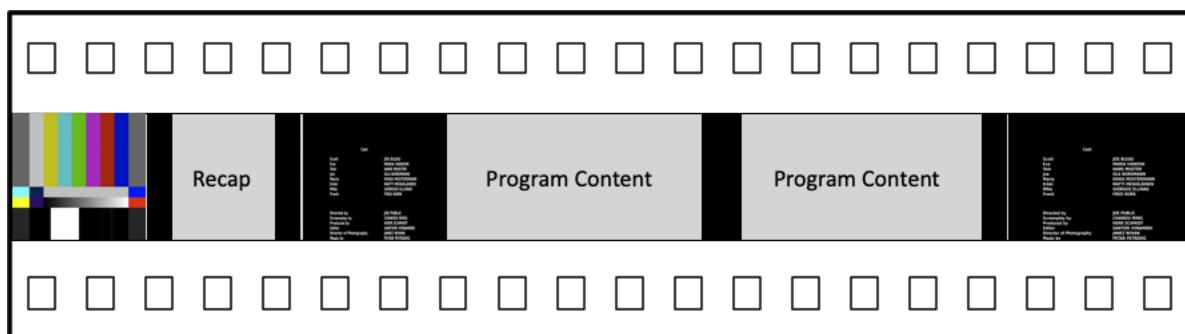
El contenido es las partes de la serie o película que contienen el programa o elementos relacionados. Los marcos negros, los créditos, las barras de colores, las pizarras y los logotipos de estudio no se consideran contenido. Amazon Rekognition Video puede detectar el inicio y el final de cada segmento de contenido del vídeo para que pueda encontrar el tiempo de ejecución del programa o segmentos específicos.

Los segmentos de contenido incluyen, entre otros, lo siguiente:

- Escenas de programas entre dos pausas publicitarias
- Un resumen rápido del episodio anterior al principio del vídeo
- Contenido adicional posterior al crédito
- Contenido «sin texto», como un conjunto de todas las escenas del programa que originalmente contenían texto superpuesto, pero donde el texto se ha eliminado para admitir la traducción a otros idiomas.

Una vez que Amazon Rekognition Video finalice la detección de todos los segmentos de contenido, puede aplicar conocimientos de dominio o enviarlos para su revisión humana para categorizar cada segmento. Por ejemplo, si utiliza vídeos que siempre comienzan con un resumen, podría clasificar el primer segmento de contenido como un resumen.

En el siguiente diagrama se muestran los segmentos técnicos de referencia en una serie o en la escala de tiempo de una película. Tenga en cuenta las barras de color y los créditos iniciales, los segmentos de contenido como el resumen y el programa principal, los fotogramas negros a lo largo del 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, puede detectar el inicio, el final y la duración de cada toma, así como contar 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 el [the section called “StartSegmentDetection” \(p. 712\)](#) y [the section called “GetSegmentDetection” \(p. 635\)](#). Operaciones de API 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 `StartSegmentDetection`Solicitud de API. 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 \(p. 341\)](#). Para ver código de ejemplo, consulte [Ejemplo: Detección de segmentos en un vídeo almacenado \(p. 346\)](#).

## 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 Segment de Amazon Rekognition 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 [Llamar a las operaciones de Amazon Rekognition Video \(p. 66\)](#).

### Temas

- [Inicio del análisis de segmentos \(p. 341\)](#)
- [Obtener resultados de análisis de segmentos \(p. 342\)](#)

## Inicio del análisis de segmentos

Para iniciar la detección de segmentos en una videollamada almacenada [StartSegmentDetection \(p. 712\)](#). Los parámetros de entrada son los mismos que otras operaciones de 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 \(p. 66\)](#).

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 matriz `SegmentTypes` para detectar segmentos de indicaciones técnicas o de tomas en el vídeo de entrada.

- TECHNICAL\_CUE: identifica las marcas de tiempo con precisión de fotogramas para el inicio, el final y la duración de las señales técnicas (fotogramas negros, barras de colores, créditos de apertura, créditos finales, logotipos de estudio y contenido del programa principal) detectados en un vídeo. Por ejemplo, puede utilizar indicaciones técnicas para encontrar el inicio de los créditos finales. Para obtener más información, consulte [Tomas técnicas \(p. 339\)](#).
- SHOT — Identifica el inicio, el final y la duración de un disparo. 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 \(p. 340\)](#).

## Filtrado de los resultados del análisis

Puede utilizar el parámetro de entrada `Filters` ([StartSegmentDetectionFilters \(p. 824\)](#)) para especificar la confianza de detección mínima que se devuelve en la respuesta. En `Filters`, use `ShotFilter` ([StartShotDetectionFilter \(p. 825\)](#)) para filtrar las tomas detectadas. Use `TechnicalCueFilter` ([StartTechnicalCueDetectionFilter \(p. 826\)](#)) para filtrar las indicaciones técnicas.

Para ver código de ejemplo, consulte [Ejemplo: Detección de segmentos en un vídeo almacenado \(p. 346\)](#).

## Obtener resultados de análisis de segmentos

Amazon Rekognition Video publica el estado de finalización del análisis de vídeo en un tema de Amazon Simple Notification Service. Si el análisis de vídeo es correcto, llame a [GetSegmentDetection \(p. 635\)](#) para 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 \(p. 68\)](#).

```
{
 "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

`GetSegmentDetection` devuelve la siguiente información general.

- Información de audio— La respuesta incluye metadatos de audio en una matriz, `AudioMetadata`, de [the section called “AudioMetadata” \(p. 737\)](#) objetos. 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 solo [the section called “VideoMetadata” \(p. 845\)](#) en el `videoMetadata` array. 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 segmento. 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 \(p. 68\)](#).
- Solicite información— El tipo de análisis solicitado en la convocatoria a `startSegmentDetection` se devuelve en el `SelectedSegmentTypes`.

## Segments

La información de indicaciones técnicas y de tomas detectadas en un vídeo se devuelven en una matriz, `Segments`, de objetos [the section called “SegmentDetection” \(p. 818\)](#). 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 temporal

Los códigos de tiempo de Amazon Rekognition Video están en [SMPTE](#) en que cada fotograma de vídeo tenga 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. [Fotograma de caída](#)Amazon Rekognition Video admite los casos de uso de tarifas. 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 marcos

La duración de cada segmento de vídeo también se expresa con el número de fotogramas. El campo StartFrameNumber indica el número de fotogramas al principio de un segmento de vídeo, y EndFrameNumber indica el número de fotogramas al final de un segmento de vídeo. DurationFrames indica el número total de fotogramas de un segmento de vídeo. Estos valores se calculan utilizando un índice de fotogramas que comienza por 0.

Puede utilizar el SegmentType para determinar el tipo de segmento que devuelve Amazon Rekognition Video.

- Tomas técnicas— el TechnicalCueSegmentfield es un [TechnicalCueSegment \(p. 834\)](#) objeto que contiene la confianza de detección y el tipo de señal técnica. Los tipos de indicaciones técnicas son: ColorBars, EndCredits, BlackFrames, OpeningCredits, StudioLogo, Slate, y Content.
- Disparo— el shotSegmentfield es un [ShotSegment \(p. 822\)](#) objeto que contiene la confianza de detección y un identificador para el segmento de 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",
 "Type": "SHOT"
 }
]
}
```

```
"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 código de ejemplo, consulte [Ejemplo: Detección de segmentos en un vídeo almacenado \(p. 346\)](#).

## 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 de [Análisis de un vídeo almacenado en un bucket de Amazon S3 con Java o Python \(SDK\) \(p. 72\)](#) que utiliza una cola de Amazon Simple Queue Service para obtener el estado de finalización de una solicitud de análisis de vídeo.

Para detectar segmentos de 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 \(SDK\) \(p. 72\)](#).
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.getStartTimercodeSMPTE());
 System.out.println("\tEnd time code: " +
detectedSegment.getEndTimercodeSMPTE());
 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(bucket, video);

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

por:

```
StartSegmentDetection(bucket, video);

if (GetSQSMessageSuccess()==true)
 GetSegmentDetectionResults();
```

Java V2

```
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();

 BlackFrame blackFrame = BlackFrame.builder()
 .maxPixelThreshold(0.2F)
 .minCoveragePercentage(60F)
 .build();

 StartShotDetectionFilter cueDetectionFilter =
 StartShotDetectionFilter.builder()
 .minSegmentConfidence(60F)
 .build();

 StartTechnicalCueDetectionFilter technicalCueDetectionFilter =
 StartTechnicalCueDetectionFilter.builder()
 .minSegmentConfidence(60F)
 .blackFrame(blackFrame)
 .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("Color range: " +
metaData.colorRange().toString());
 System.out.println("FrameRate: " + metaData.frameRate());
 System.out.println("Job");
 }

 List<SegmentDetection> detectedSegment =
segDetectionResponse.segments();
 }
}
```

```
String type = detectedSegment.get(0).type().toString();

if (type.contains(SegmentType.TECHNICAL_CUE.toString())) {
 System.out.println("Technical Cue");
 TechnicalCueSegment segmentCue =
detectedSegment.get(0).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.get(0).shotSegment();
 System.out.println("\tIndex " + segmentShot.index());
 System.out.println("\tConfidence: " +
segmentShot.confidence().toString());
}

long seconds = detectedSegment.get(0).durationMillis();
System.out.println("\tDuration : " + Long.toString(seconds) + " "
milliseconds);
System.out.println("\tStart time code: " +
detectedSegment.get(0).startTimercodeSMPTE());
System.out.println("\tEnd time code: " +
detectedSegment.get(0).endTimercodeSMPTE());
System.out.println("\tDuration time code: " +
detectedSegment.get(0).durationSMPTE());
System.out.println();

} while (segDetectionResponse != null &&
segDetectionResponse.nextToken() != null);

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

## 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):\n{segment['DurationMillis']}")\n print(f"\tStart Timestamp (milliseconds):\n{segment['StartTimestampMillis']}")\n print(f"\tEnd Timestamp (milliseconds):\n{segment['EndTimestampMillis']}")\n\n print(f"\tStart timecode: {segment['StartTimecodeSMPTE']}")\n print(f"\tEnd timecode: {segment['EndTimecodeSMPTE']}")\n print(f"\tDuration timecode: {segment['DurationSMPTE']}")\n\n print(f"\tStart frame number {segment['StartFrameNumber']}")\n print(f"\tEnd frame number: {segment['EndFrameNumber']}")\n print(f"\tDuration frames: {segment['DurationFrames']}")\n\n print()\n\n if "NextToken" in response:\n paginationToken = response["NextToken"]\n else:\n finished = True
```

2. En la función main, reemplace las líneas:

```
analyzer.StartLabelDetection()\nif analyzer.GetSQSMessagesSuccess()==True:\n analyzer.GetLabelDetectionResults()
```

por:

```
analyzer.StartSegmentDetection()\nif analyzer.GetSQSMessagesSuccess()==True:\n 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 \(SDK\) \(p. 72\)](#), 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.

# Tutoriales

Estos tutoriales multiservicio demuestran cómo utilizar las operaciones de API de Rekognition junto con otros AWS servicios para crear aplicaciones de ejemplo y realizar diversas tareas. La mayoría de estos tutoriales utilizan Amazon S3 para almacenar imágenes o videos. Otros servicios de uso común incluyen: AWS Lambda.

## Temas

- [Almacenamiento de datos de Amazon Rekognition con Amazon RDS y DynamoDB \(p. 354\)](#)
- [Uso de Amazon Rekognition y Lambda para etiquetar activos en un bucket de Amazon S3 \(p. 361\)](#)
- [Crear AWS aplicaciones de analizador de video \(p. 372\)](#)
- [Creación de una función de Lambda de Amazon Rekognition \(p. 373\)](#)
- [Uso de Amazon Rekognition para la verificación de identidad \(p. 380\)](#)

## Almacenamiento de datos de Amazon Rekognition con Amazon RDS y DynamoDB

Al utilizar las API de Amazon Rekognition, 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 guardar esas etiquetas detectadas en una base de datos. La aplicación de ejemplo desarrollada en este tutorial leerá imágenes de un [Amazon S3](#) cubo llame a la [sección llamada "DetectLabels" \(p. 583\)](#) en estas imágenes y almacena las etiquetas resultantes en una base de datos. La aplicación almacenará 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.

Usarás el [AWS SDK para Python](#) en este tutorial. También puede ver la [AWS Examples del SDK de documentación](#) en [GitHub](#) para obtener más tutoriales de Python.

## Temas

- [Requisitos previos \(p. 354\)](#)
- [Obtención de etiquetas para imágenes en un depósito de Amazon S3 \(p. 355\)](#)
- [Creación de una tabla de Amazon DynamoDB \(p. 356\)](#)
- [Carga de datos en DynamoDB \(p. 357\)](#)
- [Creación de una base de datos MySQL en Amazon RDS \(p. 358\)](#)
- [Carga de datos en una tabla MySQL de Amazon RDS \(p. 359\)](#)

## Requisitos previos

Antes de comenzar este tutorial, deberá instalar Python y completar los pasos necesarios para [Configurar Python AWS SDK](#). Más allá de esto, asegúrese de que dispone de:

Creación de una cuenta de AWS y un rol de IAM

Se ha instalado el SDK de Python (Boto3)

Configuró correctamente suAWS credenciales de acceso

El bucket de Amazon S3 creado lo llenó de imágenes

Creación de una instancia de base de datos, si utiliza RDS para almacenar datos

## Obtención de etiquetas para imágenes en un depósito de Amazon S3

Empieza escribiendo una función que tomará el nombre de una imagen en tu depósito de Amazon S3 y recuperará esa imagen. Esta imagen se mostrará para confirmar que las imágenes correctas se están pasando a una llamada a [the section called “DetectLabels” \(p. 583\)](#) que también figura en la función.

1. Busca el depósito de Amazon S3 que te gustaría usar y anota su nombre. Realizarás llamadas a este depósito de Amazon S3 y leerás las imágenes que contiene. Asegúrate de que tu depósito contenga algunas imágenes para pasar a [the section called “DetectLabels” \(p. 583\)](#).
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 conectado al recurso de Amazon S3, puede acceder al bucket proporcionando el método Bucket con el nombre del bucket de Amazon S3. Después de conectarse al bucket de Amazon S3, recupera imágenes del bucket mediante el método Object. Al utilizar 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, proporcione su región al parámetro region\_name. Pasará el nombre del bucket de Amazon S3 y el nombre de la imagen a [the section called “DetectLabels” \(p. 583\)](#), que devuelve las etiquetas de la imagen correspondiente. Después de seleccionar solo las etiquetas de la respuesta, se devuelven 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
```

- Guarda este código en un archivo llamado get\_images.py.

## Creación de una tabla de Amazon DynamoDB

El siguiente código utiliza Boto3 para conectarse a DynamoDB y utiliza DynamoDBCreateTablemétodo para crear una tabla denominada Images. La tabla tiene una clave principal compuesta que consta de una clave de partición denominada Imagen y una clave de ordenación denominada Labels. La clave Imagen contiene el nombre de la imagen, mientras que la clave Etiquetas almacena las etiquetas asignadas a esa imagen.

```
import boto3

def create_new_table(dynamodb=None):
 dynamodb = boto3.resource(
 'dynamodb')
 # Table definatoin
 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 en DynamoDB

Ahora que se ha creado la base de datos de DynamoDB y tiene una función para obtener etiquetas para imágenes, puede almacenar las etiquetas en DynamoDB. El siguiente código recupera todas las imágenes de un depósito de S3, obtiene etiquetas para ellas y almacena los datos en DynamoDB.

1. Tendrá que escribir el código para cargar los datos en DynamoDB. Una función de denominada `get_image_names` se utiliza para conectarse a su depósito de Amazon S3 y devuelve los nombres de todas las imágenes del bucket en forma de lista. Pasarás esta lista `already_image_from_S3`, que se importa desde `elget_images.py` archivo que has creado.

```
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 `read_image_from_S3` función que creamos anteriormente devolverá el nombre de la imagen que se está procesando y el diccionario de etiquetas asociado a esa imagen. Una función de denominada `find_values` se utiliza para obtener solo las etiquetas de la respuesta. El nombre de la imagen y sus etiquetas están listos para cargarse a 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. Usarás una tercera función, llamada `load_data`, para cargar las imágenes y etiquetas en la tabla de DynamoDB que creó.

```
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 hemos definido 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 usar[the section called “DetectLabels” \(p. 583\)](#)para generar etiquetas para sus imágenes y almacenarlas en una instancia de DynamoDB. Asegúrese de eliminar todos los recursos que creó mientras revisa este tutorial. Esto evitará que se le cobre por los recursos que no está utilizando.

## Creación de una base de datos MySQL en Amazon RDS

Antes de ir más allá, asegúrese de haber completado la[procedimiento de configuración](#)para Amazon RDS y[Creación de una instancia de base de datos MySQL](#)utilizando Amazon RDS.

En el siguiente código se utiliza el[PyMySQL](#)biblioteca y su instancia de base de datos de Amazon RDS. Crea una tabla para contener los nombres de las imágenes y las etiquetas asociadas a esas imágenes. Amazon RDS recibe comandos para crear tablas e insertar datos en tablas. Para utilizar Amazon RDS, debe conectarse al host de Amazon RDS mediante su nombre de host, nombre de usuario y contraseña. Se conectará a Amazon RDS proporcionando estos argumentos a la[connect](#)función y creación de una instancia de un cursor.

1. En el siguiente código, sustituya el valor del host por el extremo de host de Amazon RDS y sustituya el valor del usuario por el nombre de usuario maestro asociado a la 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 en las que insertar datos de imagen y etiquetas. Para ello, ejecute y confirme una consulta de creación. El siguiente código crea una base de datos. Ejecuta 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, pasará primero el comando use 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, llamada `image_id` 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 en una tabla MySQL de Amazon RDS

Después de 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. Connect a su depósito de Amazon S3 y recupere los nombres de todas las imágenes del bucket. Estos nombres de imagen se pasarán a la función `read_image_from_s3` que creaste anteriormente para obtener las etiquetas de todas tus imágenes. El siguiente código se conecta a su depósito de Amazon S3 y devuelve una lista de todas las imágenes del depósito.

```
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 sección titulada “[DetectLabels](#)” (p. 583)La API contiene algo más que las etiquetas, así que escribe una función para extraer solo los valores de las etiquetas. La siguiente función devuelve una lista llena de solo 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
```

- Necesitará una función para insertar los nombres de las imágenes y las etiquetas en la tabla. La siguiente función ejecuta una consulta de inserción e inserta cualquier par de nombres y etiquetas de imagen.

```
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!")
```

- Por último, debe ejecutar las funciones definidas anteriormente. En el siguiente código, se recopilan los nombres de todas las imágenes del depósito y se proporcionan a la función que llama[the section called “DetectLabels” \(p. 583\)](#). Posteriormente, 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, en 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 correctamente DetectLabels para generar etiquetas para sus imágenes y almacenarlas en una base de datos MySQL mediante Amazon RDS. Asegúrese de eliminar todos los recursos que creó mientras revisa este tutorial. Esto impedirá que se le cobre por los recursos que no utiliza.

Para obtener más información AWSEjemplos de varios servicios, consulte AWSEjemplos del SDK de documentación [Repositorio de GitHub](#).

# Uso de Amazon Rekognition y Lambda para etiquetar activos en un bucket de Amazon S3

En este tutorial, va a crear un AWS Lambda que etiqueta automáticamente los activos digitales ubicados en un bucket de Amazon S3. La función Lambda lee todos los objetos de un bucket de Amazon S3 determinado. Para cada objeto del depósito, pasa la imagen al servicio Amazon Rekognition para generar una serie de etiquetas. Cada etiqueta se utiliza para crear una etiqueta que se aplica a la imagen. Después de ejecutar la función Lambda, crea automáticamente etiquetas basadas en todas las imágenes de un depósito de Amazon S3 determinado y las aplica a las imágenes.

Por ejemplo, supongamos que ejecuta la función 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. <a href="#">Learn more</a> |          |
| 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 AWS Capa gratuita. Cuando haya terminado con el tutorial, le recomendamos que termine los recursos que haya creado durante el tutorial para que no se le cobre.

En este tutorial se utiliza AWSSDK for Java versión 2. Consulte el [AWS Documentación Ejemplos del SDK](#) [repositorio de GitHub](#) para obtener tutoriales adicionales de Java V2.

## Temas

- [Requisitos previos \(p. 361\)](#)
- [Configurar el rol Lambda de IAM \(p. 362\)](#)
- [Creación del proyecto \(p. 362\)](#)
- [Escribir el código \(p. 364\)](#)
- [Empaquetar el proyecto \(p. 371\)](#)
- [Implemente la función Lambda \(p. 371\)](#)
- [Probar el método Lambda \(p. 371\)](#)

## Requisitos previos

Antes de comenzar, debe completar los pasos en [Configuración del AWSSDK para Java](#). A continuación, asegúrese de que dispone de lo siguiente:

- Java 1.8 JDK.
- Maven 3.6 o superior.
- Un [Amazon S3](#) cubo con 5-7 imágenes de naturaleza en él. La función Lambda lee estas imágenes.

## Configurar el rol Lambda de IAM

En este tutorial se utilizan los servicios de Amazon Rekognition y Amazon S3. Configuración del soporte Lambda para tener políticas que le permitan invocar estos servicios desde una función Lambda.

Para configurar el rol

1. Inicie sesión en la AWS Management Console y abra la consola de IAM en <https://console.aws.amazon.com/iam/>.
2. En el panel de navegación, seleccione Roles y luego seleccione Crear rol.
3. Elija AWS Service y luego seleccione Lambda.
4. Elija la pestaña Permissions (Permisos).
5. Search for AWSLambdaBasicExecutionRole.
6. Elija Etiquetas siguientes.
7. Elija Review.
8. Asigne un nombre al rol de soporte Lambda.
9. Elija Create role (Crear rol).
10. Elija el soporte Lambda para ver la página de descripción general.
11. Seleccione Attach policies (Asociar políticas).
12. Elija AmazonRekognitionFullAccess en la lista de políticas.
13. Elija Asociar política.
14. Search for AmazonS3FullAccess y luego seleccione Asociar política.

## Creación del proyecto

Cree un nuevo proyecto Java y, a continuación, configure Maven pom.xml con la configuración y las dependencias necesarias. Asegúrese de que el archivo pom.xml tiene un aspecto similar al siguiente:

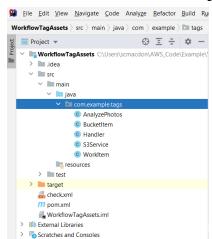
```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/
maven-4.0.0.xsd">
<modelVersion>4.0.0</modelVersion>
<groupId>org.example</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

Usar AWS Lambda API Java en tiempo de ejecución para crear la clase Java que define la función Lambda. En este ejemplo, hay una clase Java para la función Lambda denominada Handler (Controlador): y clases adicionales necesarias para este caso de uso. En la figura siguiente se muestran las clases Java del proyecto. Observe que todas las clases de Java se encuentran en un paquete denominado com.amazonaws.example.tags.



Cree las siguientes clases Java para el código:

- Handler (Controlador): utiliza la API en tiempo de ejecución de Java de Lambda y realiza el caso de uso descrito en este AWS "Hello, World!" La lógica de aplicación que se ejecuta se encuentra en el método handleRequest.
- Servicio S3: utiliza la API de Amazon S3 para realizar operaciones de S3.
- Analizar fotos: utiliza la API de Amazon Rekognition para analizar las imágenes.
- Artículo de cubo: define un modelo que almacena información de bucket de Amazon S3.
- Elemento de trabajo: define un modelo que almacena datos de Amazon Rekognition.

## Clase de controlador

Este código Java representa el Handler (Controlador): clase. La clase lee una marca que se pasa a la función Lambda. La clase s3service.listBucket devuelve unLista objeto donde cada elemento es un valor de cadena que representa la clave de objeto. Si el valor del indicador es verdadero, las etiquetas se aplican iterando por la lista y aplicando etiquetas a cada objeto llamando al S3 service.tag Assets método. Si el valor de la marca es falso, entonces S3 service.DeleteTagfrom Object se invoca que elimina las etiquetas. Además, tenga en cuenta que puede registrar mensajes en los registros de Amazon CloudWatch mediante unLambdaLogger objeto.

### Note

Asegúrese de asignar el nombre del depósito 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 S3

La clase siguiente utiliza la API de Amazon S3 para realizar operaciones de S3. Por ejemplo, el método `getObjectBytes` devuelve una matriz de bytes que representa la imagen. Del mismo modo, el método `listBucketObjects` devuelve un lista de objetos donde 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);
 }
}
```

## Clase Analizar fotos

El siguiente código Java representa elAnalizar fotosclase. 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 ;
 }
}
```

## Clase BucketItem

El siguiente código Java representa el Artículo de cubo que almacena 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 ;
}
}
```

## Clase WorkItem

El siguiente código Java representa el elemento de trabajo clase.

```
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

Empaque el proyecto en un archivo.jar (JAR) mediante el siguiente comando de Maven.

```
mvn package
```

El archivo JAR se encuentra en eltargetcarpeta (que es una carpeta secundaria de la carpeta del proyecto).

Name	Date modified	Type	Size
classes	3/31/2021 9:57 AM	File	
generated-sources	3/30/2021 1:26 AM	File	
generated-test-sources	3/30/2021 1:26 AM	File	
maven-archiver	3/30/2021 1:20 PM	File	
maven-status	3/30/2021 1:20 PM	File	
plexus-buildapi	3/30/2021 1:20 PM	File	
checkstyle-cachefile	3/31/2021 9:51 AM	File	1 KB
checkstyle-checker.xml	3/31/2021 9:51 AM	XML Document	1 KB
checkstyle-result.xml	3/31/2021 9:51 AM	XML Document	1 KB
original-WorkflowTagsesito-1.0-SNAPSHOT.jar	3/31/2021 9:47 AM	Executable Jar File	11 KB
NewWorkflowesito-1.0-SNAPSHOT.jar	3/31/2021 9:47 AM	Executable Jar File	12,869 KB
NewWorkflowesito-1.0-SNAPSHOT-shaded.jar	3/31/2021 9:47 AM	Executable Jar File	12,869 KB

### Note

Observe el uso delmaven-shade-pluginen el archivo POM del proyecto. Este complemento es responsable de crear un JAR que contenga las dependencias necesarias. Si intenta empaquetar el proyecto sin este complemento, las dependencias necesarias no se incluyen en el archivo JAR y encontrará unClase no encontrada, excepto.

## Implemente la función Lambda

1. Abra la [consola de Lambda](#).
2. Elija Create Function (Crear función).
3. Elija Author from scratch (Crear desde cero).
4. En el navegadorInformación básica sección, introduzca el nombre como nombre.
5. En el navegadorRuntime (Tiempo de ejecución):, eligeJava 8.
6. ElegirUtilizar un rol existente y luego seleccionesoporte lambda(el rol de IAM que ha creado).
7. Elija Create function (Crear función).
8. ParaCode entry type, eligeCargar un archivo .zip o .jar.
9. ElegirCargary, a continuación, navegue hasta el archivo JAR que ha creado.
- 10ParaHandler (Controlador):, introduzca el nombre completo de la función, por ejemplo,com.example.tags.handler:solicitud de manejo(com.example.tagspecifica el paquete, Handler (Controlador):es la clase seguida de# y nombre del método).
- 11Elija Save (Guardar).

## Probar el método Lambda

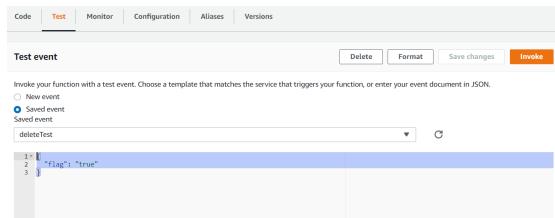
En este punto del tutorial, puede probar la función Lambda.

1. En la consola de Lambda, haga clic enPruebasy luego escriba el JSON siguiente.

```

"flag": "true"
}

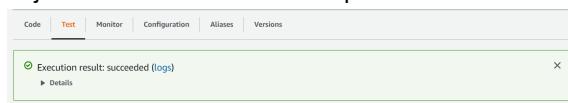
```



#### Note

Pasotruetiqueta los activos digitales y el pasofalseelimina las etiquetas.

- Elija el iconoInvocarbotón. Después de invocar la función Lambda, verá un mensaje correcto.



Enhorabuena, ha creado unAWS Lambda que aplica etiquetas automáticamente a activos digitales ubicados en un bucket de Amazon S3. Como se indica al principio de este tutorial, asegúrese de terminar todos los recursos que creó mientras revisa este tutorial para asegurarse de que no se le cobra.

Para obtener más informaciónAWSEjemplos multiservicio, consulte[AWS Documentación Ejemplos del SDK repositorio de GitHub](#).

## CrearAWSAplicaciones de analizador de vídeo

Puede crear una aplicación web Java que analice los vídeos para detectar etiquetas mediante elAWSSDK for Java versión 2. La aplicación creada en esteAWSEI tutorial le permite cargar 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 llenar un modelo de datos y, a continuación, se genera un informe y se envía por correo electrónico a un usuario específico mediante Amazon Simple Email Service.

En la siguiente ilustración se muestra un informe que se genera después de que la aplicación finalice el análisis del vídeo.

Age Range	Brand	Eye always	Eye open
1	ApplianceLow-18, High=56	BrandValue=false, Confidence=83.0723	EyeAlwaysValue=true, Confidence=55.965977
2	ApplianceLow-16, High=52	BrandValue=true, Confidence=87.7212	EyeAlwaysValue=false, Confidence=63.565166
3	ApplianceLow-14, High=52	BrandValue=false, Confidence=87.7212	EyeAlwaysValue=false, Confidence=55.389756
4	ApplianceLow-12, High=52	BrandValue=false, Confidence=81.4521	EyeAlwaysValue=false, Confidence=63.586451
5	ApplianceLow-10, High=67	BrandValue=false, Confidence=81.41662	EyeAlwaysValue=true, Confidence=65.28722
6	ApplianceLow-8, High=69	BrandValue=true, Confidence=87.18313	EyeAlwaysValue=false, Confidence=95.1523
7	ApplianceLow-6, High=69	BrandValue=true, Confidence=87.18313	EyeAlwaysValue=true, Confidence=82.2171
8	ApplianceLow-4, High=69	BrandValue=true, Confidence=87.12265	EyeAlwaysValue=true, Confidence=84.4177
9	ApplianceLow-51, High=69	BrandValue=true, Confidence=84.93304	EyeAlwaysValue=false, Confidence=94.14824
10	ApplianceLow-44, High=62	BrandValue=true, Confidence=84.8414	EyeAlwaysValue=true, Confidence=95.83134
11			EyeOpenValue=true, Confidence=98.138665

En este tutorial, va a crear una aplicación Spring Boot que invoca variosAWSservicios de . Las API de Spring Boot se utilizan para crear un modelo, distintas vistas y un controlador. Para obtener más información, consulte[Bota de Spring](#).

Este servicio utiliza lo siguiente:AWSservicios de:

- Amazon Rekognition
- Amazon S3
- Amazon SES

- [AWS Elastic Beanstalk](#)

Los servicios incluidos en este tutorial se incluyen en el AWS Capa gratuita. Le recomendamos que termine todos los recursos que cree en el tutorial cuando haya terminado con ellos para evitar que se le carguen.

## Requisitos previos

Antes de comenzar, debe completar los pasos que aparecen en [Configuración del AWSSDK para Java](#). A continuación, asegúrese de que dispone de lo siguiente:

- Java 1.8 JDK.
- Maven 3.6 o posterior.
- Un bucket de Amazon S3 denominado video [algún valor]. Asegúrese de utilizar este nombre de depósito en el código Java de Amazon S3. Para obtener más información, consulte [Creating a bucket \(Creación de un bucket\)](#).
- Un rol de IAM. Necesita esto para la Detección de rostros de videoclase que crearás. Para obtener más información, consulte [Configuración de Amazon Rekognition Video](#).
- Un tema válido de Amazon SNS. Necesita esto para la Detección de rostros de videoclase que crearás. Para obtener más información, consulte [Configuración de Amazon Rekognition Video](#).

## Procedimiento

En el curso del tutorial, va a hacer lo siguiente:

1. Crear un proyecto
2. Agregar las dependencias POM a su proyecto
3. Creación de las clases Java
4. Cree los archivos HTML
5. Cree los archivos de script
6. Empaquetar el proyecto en un archivo JAR
7. Implemente la aplicación en AWS Elastic Beanstalk

Para continuar con el tutorial, siga las instrucciones detalladas que aparecen en la [AWS Documentación Ejemplos del SDK repositorio de 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 de Java.

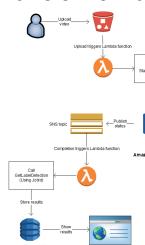
### Note

En este tutorial se utiliza AWSSDK for Java 1.x. Para obtener un tutorial utilizando Rekognition y el AWSSDK for Java versión 2, consulte la [AWS Documentación Ejemplos del SDK repositorio de 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 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 [the section called “StartLabelDetection” \(p. 701\)](#) Para comenzar la detección de etiquetas en el vídeo cargado. Para obtener información acerca del uso de Lambda para procesar notificaciones de eventos desde un bucket de Amazon S3, consulte [Uso de AWS Lambda con Amazon S3 Events](#).

Una segunda función Lambda se activa cuando se envía el estado de realización del análisis al tema de Amazon SNS registrado. La segunda función Lambda llama [the section called “GetLabelDetection” \(p. 626\)](#) para 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 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 [the section called “GetLabelDetection” \(p. 626\)](#). Para fines de demostración, este tutorial escribe los resultados de la detección de etiquetas en un log de CloudWatch. La función 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:

- Cree el tema de Amazon SNS y configure los permisos.
- Cree la función Lambda mediante la AWS Management Console Suscríbase al tema de Amazon SNS.
- Configure la función Lambda mediante la AWS Management Console.
- Añadir código de ejemplo a un AWS Toolkit for Eclipse proyecto y cargue en la función Lambda.
- Probar la función Lambda mediante la AWS CLI.

#### Note

Utilice la misma región de AWS 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 la función de servicio de IAM que otorga a Amazon Rekognition Video acceso a los temas de Amazon SNS. Para obtener más información, consulte [Llamar a las operaciones de Amazon Rekognition Video \(p. 66\)](#).

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 [Acceso a varios temas de Amazon SNS \(p. 71\)](#).

2. Cree un tema de Amazon SNS mediante el uso de la[Consola de Amazon SNS](#). Solo tiene que especificar el nombre del tema. Anexe AmazonRekognition al nombre del tema. Apunte el ARN del tema.

## Crear la función de Lambda

La función Lambda puede crear mediante la AWS Management Console. A continuación, utiliza un AWS Toolkit for Eclipse proyecto de para cargar el paquete de funciones de Lambda en AWS Lambda. También es posible crear la función Lambda con la 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 Create function (Crear función).
3. Elija Author from scratch (Crear desde cero).
4. En Function name (Nombre de función), introduzca un nombre para la función.
5. En Runtime (Tiempo de ejecución), elija Java 8.
6. Elija Choose or create an execution role (Seleccionar o crear un rol de ejecución).
7. En Execution role (Rol de ejecución), elija Create a new role with basic Lambda permissions (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 Basic information (Información básica).
9. Elija Create function (Crear función).

## Configure la función Lambda

Una vez que haya creado la función Lambda, puede configurarla para que la active el tema de Amazon SNS que se crea en[Creación del tema de SNS \(p. 374\)](#). También deberá ajustar los requisitos de memoria y el periodo de tiempo de espera para la función Lambda.

Para configurar la función Lambda

1. En Function Code (Código de función), escriba `com.amazonaws.lambda.demo.JobCompletionHandler` para Handler (Controlador).
  2. En Basic settings (Configuración básica), elija Edit (Editar). Se muestra el cuadro de diálogo Edit basic settings (Editar configuración básica).
    - a. Elija 1024 para Memory (Memoria).
    - b. Elija 10 segundos para Timeout (Tiempo de espera).
    - c. Elija Save (Guardar).
  3. En Designer (Diseñador), elija + Add trigger (+ Agregar desencadenador). Se muestra el cuadro de diálogo Add trigger (Aregar desencadenador).
  4. En Trigger configuration (Configuración de desencadenador) elija SNS.
- En Tema de SNS, elija el tema de Amazon SNS que ha creado en[Creación del tema de SNS \(p. 374\)](#).
5. Elija Enable trigger (Activar disparador).

6. Para añadir el disparador, elija Add (Añadir).
7. Elegir Guardar para guardar la función Lambda.

## Configurar el rol Lambda de IAM

Para llamar a las operaciones de Amazon Rekognition Video, agregue el `AmazonRekognitionFullAccess` La política administrada de AWS para el rol de Lambda de IAM. Iniciar operaciones, tales como [the section called "StartLabelDetection" \(p. 701\)](#), también requiere que el rol de servicio de IAM que Amazon Rekognition Video utiliza para obtener acceso al tema de Amazon SNS.

Para configurar el rol

1. Inicie sesión en la AWS Management Console y abra la consola de IAM en <https://console.aws.amazon.com/iam/>.
2. Seleccione Roles (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 \(p. 375\)](#).
4. Elija la pestaña Permissions (Permisos).
5. Seleccione Attach policies (Asociar políticas).
6. Elija `AmazonRekognitionFullAccess` en la lista de políticas.
7. Elija Asociar política.
8. Nuevamente, elija el rol de ejecución.
9. Elija Add inline policy (Agregar política insertada).
10. Seleccione la pestaña JSON.
11. Reemplace la política existente por la siguiente política. Reemplazar `servicerole` con el rol de servicio de IAM que creó en [Creación del tema de SNS \(p. 374\)](#).

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Sid": "mysid",
 "Effect": "Allow",
 "Action": "iam:PassRole",
 "Resource": "arn:aws:iam::123456789012:role/servicerole"
 }
]
}
```

12. Elija Review policy (Revisar política).
13. En Name\* (Nombre\*), escriba un nombre para la política.
14. Elija Create Policy (Crear política).

## Crear la AWS Toolkit for Eclipse Proyecto Lambda

Cuando se activa la función Lambda, el código siguiente obtiene el estado de realización del tema de Amazon SNS y llama a [the section called "GetLabelDetection" \(p. 626\)](#) para obtener los resultados del análisis. A continuación, se escribe una lista de las etiquetas detectadas, así como el recuento de estas, en un registro de CloudWatch. La función Lambda debería almacenar los resultados del análisis de vídeo para su uso posterior.

## Para crear elAWS Toolkit for EclipseProyecto Lambda

### 1. Creación de unAWS Toolkit for EclipseAWSProyecto Lambda.

- En Project name (Nombre del proyecto), escriba un nombre de proyecto de su elección.
- En Class Name: (Nombre de clase:), escriba JobCompletionHandler.
- En Input type: (Tipo de entrada), elija SNS Event (Evento de SNS).
- Deje el resto de los campos sin modificar.

### 2. En el navegadorProyecto Eclipseexplorer, abra el método del controlador Lambda generado (JobCompletionHandler.java) y reemplace 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);
 }
 }
 } catch (Exception e) {
 logger.error(e.getMessage());
 }
 }
}
```

```
 }

 } 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(paginationToken);

 labelDetectionResult = rek.getLabelDetection(labelDetectionRequest);

 paginationToken = labelDetectionResult.getNextToken();
 VideoMetadata videoMetaDataAdapter = 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 Fix project set up... (Corregir configuración del proyecto...).
  - Elija la última versión del archivo de Amazon Rekognition.
  - Elija OK (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 Upload function to AWS Lambda.
6. En la página Select Target Lambda Function, elija la región de AWS que desea utilizar.
7. Elija Choose an existing lambda function (Elegir una función de Lambda existente) y seleccione la función de Lambda que ha creado en [Crear la función de Lambda \(p. 375\)](#).
8. Elija Next (Siguiente). Se muestra el cuadro de diálogo Function Configuration (Configuración de función).
9. En IAM Role (Rol de IAM), elija el rol de IAM que creó en [Crear la función de Lambda \(p. 375\)](#).
10. ElegirAcabado, y la función Lambda se carga enAWS.

## Probar la función de Lambda

Utilice lo siguienteAWS CLIpara probar la función Lambda iniciando el análisis de detección de etiquetas de un vídeo. Una vez finalizado el análisis, la función Lambda se activa. Confirme que el análisis se realizó correctamente; para ello, compruebe los logs de CloudWatch Logs

### 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 obtener instrucciones, consulte[Carga de objetos en Amazon S3en laAmazon Simple Storage Service.](#)
2. Ejecute el siguiente comando de AWS CLI para comenzar la detección de 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:

- Cambiobucketnameyvideofileal nombre de bucket de Amazon S3 y nombre de archivo del vídeo donde desea detectar etiquetas.
  - CambioTopicARNal ARN del tema de Amazon SNS que ha creado en[Creación del tema de SNS \(p. 374\)](#).
  - CambioRoleARNal ARN del rol de IAM que creó en[Creación del tema de SNS \(p. 374\)](#).
  - Cambie Region a la región de AWS que está 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 el icono <https://console.aws.amazon.com/cloudwatch/> consola de .
5. Cuando finalice el análisis, aparecerá una entrada de registro para la función Lambda en la Grupo de logs.
6. Elija la función Lambda para ver los flujos de registros.
7. Elija el flujo de logs más reciente para ver las entradas de log realizadas por la función de Lambda. Si la operación se realiza correctamente, la pantalla tendrá un aspecto similar al siguiente:

Time (UTC +00:00)	Message
2018-02-28	START RequestId: 47cb1472-1cc0-11e8-860a-4d00aa2ff96b Version: \$LATEST
19:48:01	Rekognition Video Operation:-----
19:48:02	Job id: "9c7301403a375a044c6dbe793d5c78d06014ee165efde083ad654b06fc59a"
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: 70 ms Billing Duration

El valor de Job id (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 de una imagen y, a continuación, comparar las caras detectadas con otras caras comparando los datos de rostros. Estos datos de cara se almacenan en contenedores del lado del servidor denominados Colecciones. Al utilizar las operaciones de detección de rostros, comparación de rostros y administración de colecciones de Amazon Rekognition, puede crear una aplicación con una solución de verificación de identidad.

En este tutorial se mostrarán dos flujos de trabajo comunes para la creación de aplicaciones que requieren verificación de identidad.

El primer flujo de trabajo implica el registro de un nuevo usuario en una colección. El segundo flujo de trabajo consiste en buscar una colección existente con el fin de iniciar sesión en un usuario que regresa.

Usarás el [AWSSDK para Python](#) para este tutorial. También puede ver la [AWS Ejemplos del SDK de documentación](#) [repositorio de GitHub](#) para obtener más tutoriales de Python.

### Temas

- [Requisitos previos \(p. 380\)](#)
- [Creación de una colección \(p. 381\)](#)
- [Registro de nuevos usuarios \(p. 382\)](#)
- [Inicio de sesión del usuario existente \(p. 387\)](#)

## Requisitos previos

Antes de comenzar este tutorial, tendrá que instalar Python y completar los pasos necesarios para [configurar Python AWSSDK](#). Más allá de esto, asegúrese de que dispone de:

- Ha creado un AWS cuenta y un rol de IAM
- Se ha instalado el SDK de Python (Boto3)
- Configuró correctamente su AWS credenciales de acceso
- Se ha creado un bucket de Amazon Simple Storage Service y cargué una imagen que deseas utilizar como ID para la verificación de identidad.
- Ha seleccionado una segunda imagen para que sirva de imagen de destino para la verificación de identidad.

## Creación de una colección

Antes de poder registrar un nuevo usuario en una colección o buscar un usuario en una colección, debe tener 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 las caras detectadas.

### Crear la recopilación de

Empezarás escribiendo una función que crea una colección para que la utilice tu aplicación. Amazon Rekognition almacena información sobre las caras 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 tiene que crear una colección mediante `CreateCollection`.

1. Seleccione un nombre para la colección que le gustaría crear. En el siguiente código, sustituya el valor `collection_id` con el nombre de la colección que desea crear y sustituya el valor `region` con el nombre de la región definido en las credenciales de usuario. Puede utilizar `Tags` para aplicar cualquier etiqueta que desee a la colección, aunque no es obligatorio. La `CreateCollection` devolverá información sobre la colección que ha creado, incluido el Arn de la colección. Anote el Arn que recibe como resultado de 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. Copia la colección Arn.

Ahora que se ha creado Rekognition Collection, puede almacenar información facial e identificadores en esa colección. También podrás comparar rostros con la información almacenada para su verificación.

## Registro de nuevos usuarios

Querrás poder registrar nuevos usuarios y añadir su información a una colección. El proceso de registro de un nuevo usuario suele incluir los pasos siguientes:

### Llame a laDetectFacesOperación

Escriba el código para comprobar la calidad de la imagen facial a través delDetectFaces. Usarás elDetectFacespara determinar si una imagen capturada por la cámara es adecuada para su procesamiento por elsearchFacesByImage. La imagen debe contener solo una cara. Proporcionarás un archivo de imagen de entrada local a laDetectFacesoperación y reciba detalles de los rostros detectados en la imagen. El siguiente código de ejemplo proporciona la imagen de entrada paraDetectFacesy, a continuación, comprueba si solo se ha detectado una cara en la imagen.

1. En el siguiente ejemplo de código, sustituyaphotocon el nombre de la imagen de destino en la que quieras detectar rostros. También tendrá que sustituir el valor dendregioncon el nombre de la región asociada a tu 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.

## Llame a laCompareFacesOperación

Su aplicación deberá poder registrar nuevos usuarios en una colección y confirmar la identidad de los usuarios que regresan. Creará las funciones utilizadas para registrar primero a un nuevo usuario. Empezarás usando elCompareFacespara comparar una imagen de entrada/destino local del usuario y un ID/imagen almacenada. Si se detecta una coincidencia entre la cara en ambas imágenes, puede buscar en la colección para ver si el usuario se ha registrado en ella.

Empieza escribiendo una función que compara una imagen de entrada con la imagen de ID que has almacenado en tu depósito de Amazon S3. En el siguiente ejemplo de código, u=tendrá que proporcionar la imagen de entrada usted mismo, que debe capturarse después de utilizar algún tipo de detector de vida. También tendrá que pasar el nombre de una imagen almacenada en su bucket de Amazon S3.

1. Sustituir el valor debucketcon el nombre del bucket de Amazon S3 que contiene el archivo de origen. También tendrá que sustituir el valor desource\_filecon el nombre de la imagen de origen que está utilizando. Sustituir el valor detarget\_filecon el nombre del archivo de destino que ha proporcionado. Sustituir el valor dendregioncon el nombre de laregiondefinidas en las credenciales de usuario.

También querrá especificar un nivel de confianza mínimo en la coincidencia que se devuelve en la respuesta, utilizando elsimilarityThresholdargumento. Las caras detectadas solo se devolverán en elFaceMatchesarray si la confianza está por encima de este umbral. Tu elegidosimilarityThresholddebe reflejar la naturaleza de su caso de uso específico. Cualquier caso de uso que implique aplicaciones de seguridad críticas debe utilizar 99 como 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 la cara coincidente y el nivel de confianza.

## Llame a la `SearchFacesByImage` Operación

Si el nivel de confianza del `CompareFaces` la operación está por encima de la elegida `SimilarityThreshold`, querrás buscar en tu colección una cara que coincida con la imagen de entrada. Si se encuentra una coincidencia en tu colección, significa que es probable que el usuario ya esté registrado en la colección y no es necesario registrar un nuevo usuario en tu colección. Si no hay ninguna coincidencia, puedes registrar al nuevo usuario en tu colección.

1. Empieza escribiendo el código que invocará `SearchFacesByImage`. La operación tomará un archivo de imagen local como argumento y, a continuación, buscará `enCollection` para una cara que coincida con las caras detectadas más grandes de la imagen proporcionada.

En el siguiente ejemplo de código, cambie el valor de `collectionId` a la colección en la que se desea realizar una búsqueda. Sustituir el valor de `region` con el nombre de la región asociada a tu cuenta. También tendrá que sustituir el valor de `photo` con el nombre del archivo de entrada. También querrá especificar un umbral de similitud reemplazando el valor de `threshold` con 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 seguir los siguientes pasos. En este caso, solo tiene que permitir que el usuario tenga acceso a la aplicación.

## Llame a la `IndexFaces` Operación

Suponiendo que no se haya encontrado ninguna coincidencia en la colección que ha buscado, querrá añadir la cara del usuario a su colección. Haces esto llamando a `IndexFaces`. Cuando llamas a `IndexFaces`, Amazon Rekognition extrae los rasgos faciales de una cara identificada en la imagen de entrada y almacena los datos en la colección especificada.

1. Comience escribiendo el código para llamar `IndexFaces`. Sustituir el valor de `image` con el nombre del archivo local que desea utilizar como imagen de entrada para la operación `IndexFaces`. También tendrá que sustituir el valor de `photo_name` con el nombre deseado para la imagen de entrada. Asegúrese de sustituir el valor de `collection_id` con el ID de la colección que creó anteriormente. A continuación, sustituya el valor de `region` con el nombre de la región asociada a tu cuenta. También querrá especificar un valor para el `MaxFaces` parámetro de entrada, 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 `IndexFaces`, como el ID de cara asignado a la persona de la imagen. En la siguiente sección se examinará cómo guardar estos datos. Copiar los devueltos `FaceId`, `ImageId`, y `Confidence` valores antes de proceder.

## Almacenar datos de imagen y FaceID en Amazon S3 y Amazon DynamoDB

Una vez obtenido el Face ID de la imagen de entrada, los datos de imagen se pueden guardar en Amazon S3, mientras que los datos de cara y la URL de la imagen se pueden introducir en una base de datos como DynamoDB.

1. Escriba el código para cargar la imagen de entrada en la base de datos de Amazon S3. En el ejemplo de código siguiente, sustituya el valor `debucket` con el nombre del bucket en el que desea cargar el archivo y, a continuación, sustituya el valor `defile_name` con el nombre del archivo local que desea almacenar en el bucket de Amazon S3. Proporcione un nombre clave que identificará el archivo en el depósito de Amazon S3 reemplazando el valor `dkey_name` con un nombre al que te gustaría dar el archivo de imagen. El archivo que desea cargar es el mismo que se definió en ejemplos de código anteriores, que es el archivo de entrada que utilizó para `IndexFaces`. Por último, sustituya el valor `dregion` con el nombre de la región asociada a tu cuenta.

```
import boto3
import logging
from botocore.exceptions import ClientError

store local file in S3 bucket
bucket = "bucket-name"
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. Guarda y ejecuta el ejemplo de código correspondiente para cargar la imagen de entrada en Amazon Amazon S3.
3. También querrá guardar el Face ID devuelto en una base de datos. Esto se puede hacer creando una tabla de base de datos de DynamoDB y, a continuación, cargando el Face ID 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 siguiente código, sustituya el valor `dregion` con el valor de la región asociada a su cuenta de. También tendrá que sustituir el valor `dedatabase_name` con el nombre que te gustaría dar a la tabla 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 de procedimiento para crear la tabla.

5. Despu s de crear la tabla, puede cargar el FaceID devuelto en ella. Para ello, establecer  una conexi n con la tabla con la funci n Tabla y, a continuaci n, utilizar  elput\_item para cargar los datos.

En el siguiente ejemplo de c digo, sustituya el valor debucket con el nombre del bucket de que contiene la imagen de entrada que ha cargado a Amazon S3. Tambi n tendr  que sustituir el valor de file\_name con el nombre del archivo de entrada que carg  en el bucket de Amazon S3 y el valor de key\_name con la clave que utiliz  anteriormente para identificar el archivo de entrada. Por ltimo, sustituya el valor de region con el nombre de la regi n asociada a tu cuenta. Estos valores deben coincidir con los proporcionados en el paso 1.

La AddDBEntryalmacena los valores FaceID, ImageID y Confidence asignados a una cara de una colecci n. Proporcione a la funci n siguiente los valores que guard  durante el paso 2 del procedimiento IndexFacessecci n.

```
import boto3
from pprint import pprint
from decimal import Decimal
import json

The local file that was stored in S3 bucket
bucket = "s3-bucket-name"
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 c digo correspondiente para almacenar los datos de Face ID devueltos en una tabla.

## Inicio de sesi n del usuario existente

Despu s de que un usuario se haya registrado en una colecci n, se puede autenticar a su devoluci n mediante elSearchFacesByImage. Tendr  que obtener una imagen de entrada y, a continuaci n, comprobar la calidad de la imagen de entrada medianteDetectFaces. Esto determina si se ha utilizado una imagen adecuada antes de ejecutar elSearchFacesbyImage.

## Llame a la operación DetectFaces

1. Usarás el `DetectFaces` para comprobar la calidad de la imagen facial y determinar si una imagen capturada por la cámara es adecuada para su procesamiento por `elsearchFacesByImage`. La imagen de entrada debe contener una sola cara. El siguiente ejemplo de código toma una imagen de entrada y la proporciona al `DetectFaces`.

En el siguiente ejemplo de código, sustituya el valor `dephoto` con el nombre de la imagen de destino local y sustituya el valor `deregion` con el nombre de la región asociada a tu 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.

## Llame a la operación SearchFacesByImage

1. Escriba el código para comparar la cara detectada con las caras de la colección con `elSearchFacesByImage`. Usarás el código que se muestra en la sección Registro de nuevos usuarios que se procede y proporcionarás la imagen de entrada al `SearchFacesByImage`.

En el siguiente ejemplo de código, cambie el valor `decollectionId` a la colección que desea buscar. También cambiará el valor `debucket` al nombre de un bucket de Amazon S3 y el valor `defileName` a un archivo de imagen de ese depósito. Sustituir el valor `deregion` con el nombre de la

región asociada a tu cuenta. También querrá especificar un umbral de similitud reemplazando el valor `dethreshold` con un percentil elegido.

```
import boto3

bucket = 'bucket-name'
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 FacelID devuelto y el nivel de confianza

Ahora puede buscar información sobre el FacelID coincidente imprimiendo elementos de respuesta como los atributos FacelID, 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']))
```

# Ejemplos de código para Amazon Rekognition

Los siguientes ejemplos de código muestran cómo utilizar Amazon Rekognition con unAWSkit de desarrollo de software (SDK). Los ejemplos de código de este capítulo están destinados a complementar los ejemplos de código que se encuentran en el resto de esta guía.

Si desea obtener una lista completa de las guías para desarrolladores de SDK de AWS y ejemplos de código, incluida ayuda para empezar e información sobre versiones anteriores, consulte [Uso de Rekognition con unAWSSDK \(p. 15\)](#).

## Ejemplos de código

- [Ejemplos de servicios combinados para Amazon Rekognition \(p. 391\)](#)
  - Detectar EPP en imágenes con Amazon Rekognition mediante unAWSSDK (p. 391)
  - Detectar rostros en una imagen mediante unAWSSDK (p. 392)
  - Detección objetos en imágenes con Amazon Rekognition mediante unAWSSDK (p. 393)
  - Detección personas y objetos en un vídeo con Amazon Rekognition mediante un SDK de AWS (p. 394)
  - Guardar EXIF y otra información de imagen mediante unAWSSDK (p. 395)
- [Ejemplos de uso de Amazon Rekognition \(p. 396\)](#)
  - Cree una colección de Amazon Rekognition y encuentre caras en ella mediante unAWSSDK (p. 396)
  - Detecte y muestre elementos en imágenes con Amazon Rekognition mediante unAWSSDK (p. 403)
  - Detecte la información de los vídeos mediante Amazon Rekognition y elAWSSDK (p. 412)
- [Ejemplos de API para Amazon Rekognition \(p. 430\)](#)
  - Comparar los rostros de una imagen con una imagen de referencia con Amazon Rekognition mediante unAWSSDK (p. 430)
  - Crear una colección de Amazon Rekognition utilizando unAWSSDK (p. 435)
  - Eliminar una colección de Amazon Rekognition mediante unAWSSDK (p. 437)
  - Eliminar caras de una colección de Amazon Rekognition mediante unAWSSDK (p. 439)
  - Describir una colección de Amazon Rekognition mediante unAWSSDK (p. 442)
  - Detectar rostros de una imagen con Amazon Rekognition mediante unAWSSDK (p. 445)
  - Detectar etiquetas de una imagen con Amazon Rekognition mediante unAWSSDK (p. 450)
  - Detección etiquetas de moderación en una imagen con Amazon Rekognition mediante unAWSSDK (p. 454)
  - Detectar texto de una imagen con Amazon Rekognition mediante unAWSSDK (p. 457)
  - Obtenga información sobre celebridades con Amazon Rekognition utilizando unAWSSDK (p. 460)
  - Indexar caras de una colección de Amazon Rekognition mediante unAWSSDK (p. 461)
  - Enumerar colecciones de Amazon Rekognition utilizando unAWSSDK (p. 465)
  - Enumerar caras de una colección de Amazon Rekognition utilizando unAWSSDK (p. 468)
  - Reconoce a las celebridades de una imagen con Amazon Rekognition usando unAWSSDK (p. 471)
  - Buscar rostros en una colección de Amazon Rekognition mediante unAWSSDK (p. 474)

- Buscar rostros en una colección de Amazon Rekognition en comparación con una imagen de referencia mediante unAWSSDK (p. 477)

## Ejemplos de servicios combinados para Amazon Rekognition

Los siguientes ejemplos de código muestran cómo utilizar Amazon Rekognition conAWSSDK.

### Ejemplos

- Detectar EPP en imágenes con Amazon Rekognition mediante unAWSSDK (p. 391)
- Detectar rostros en una imagen mediante unAWSSDK (p. 392)
- Detección objetos en imágenes con Amazon Rekognition mediante unAWSSDK (p. 393)
- Detección personas y objetos en un vídeo con Amazon Rekognition mediante un SDK de AWS (p. 394)
- Guardar EXIF y otra información de imagen mediante unAWSSDK (p. 395)

## Detectar EPP en imágenes con Amazon Rekognition mediante unAWSSDK

Los siguientes ejemplos de código muestran cómo crear una aplicación que utiliza Amazon Rekognition para detectar equipos de protección personal (EPP) en imágenes.

### Java

#### SDK para Java 2.x

Muestra cómo crear unAWS Lambdafunción que detecta imágenes con equipo de protección personal.

Para ver el código fuente completo y las instrucciones de configuración y ejecución, consulte el ejemplo completo en [GitHub](#).

#### Servicios utilizados en este ejemplo

- DynamoDB
- Amazon Rekognition
- Amazon S3
- Amazon SES

### JavaScript

#### SDK para JavaScript V3

Muestra cómo utilizar Amazon Rekognition con elAWS SDK for JavaScriptpara crear una aplicación para detectar equipos de protección personal (EPP) en imágenes ubicadas en un bucket de Amazon Simple Storage Service (Amazon S3). La aplicación guarda los resultados en una tabla de Amazon DynamoDB y envía al administrador una notificación por correo electrónico con los resultados mediante Amazon Simple Email Service (Amazon SES).

Aprenda a:

- Cree un usuario no autenticado mediante Amazon Cognito.
- Analiza imágenes para EPI mediante Amazon Rekognition.
- Verificar una dirección de email de Amazon SES.
- Actualizar una tabla de DynamoDB con los resultados.
- Enviar una notificación por email mediante Amazon SES.

Para ver el código fuente completo y las instrucciones de configuración y ejecución, consulte el ejemplo completo en [GitHub](#).

Servicios utilizados en este ejemplo

- DynamoDB
- Amazon Rekognition
- Amazon S3
- Amazon SES

Si desea obtener una lista completa de las guías para desarrolladores de SDK de AWS y ejemplos de código, incluida ayuda para empezar e información sobre versiones anteriores, consulte [Uso de Rekognition con unAWSSDK \(p. 15\)](#).

## Detector rostros en una imagen mediante unAWSSDK

En el siguiente ejemplo de código, se muestra cómo:

- Guarde una imagen en un bucket de Amazon Simple Storage Service (Amazon S3).
- Utiliza Amazon Rekognition (Amazon Rekognition) para detectar detalles faciales, como rango de edad, sexo y emoción (sonrientes, etc.).
- Muestra esos detalles.

Rust

SDK para Rust

Note

Esta documentación es para un SDK en versión preliminar. El SDK está sujeto a cambios y no se debe utilizar en producción.

Guardar la imagen en un bucket de Amazon Simple Storage Servicesubidasprefijo, usa Amazon Rekognition para detectar detalles faciales, como rango de edad, sexo y emoción (sonrientes, etc.) y mostrar esos detalles.

Para ver el código fuente completo y las instrucciones de configuración y ejecución, consulte el ejemplo completo en [GitHub](#).

Servicios utilizados en este ejemplo

- Amazon Rekognition
- Amazon S3

Si desea obtener una lista completa de las guías para desarrolladores de SDK de AWS y ejemplos de código, incluida ayuda para empezar e información sobre versiones anteriores, consulte [Uso de Rekognition con unAWSSDK \(p. 15\)](#).

## Detección objetos en imágenes con Amazon Rekognition mediante unAWSSDK

Los siguientes ejemplos de código muestran cómo crear una aplicación que utilice Amazon Rekognition para detectar objetos por categoría en imágenes.

### Java

#### SDK para Java 2.x

Muestra cómo utilizar Amazon Rekognition Java API 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 email con los resultados mediante Amazon Simple Email Service (Amazon SES).

Para ver el código fuente completo y las instrucciones de configuración y ejecución, consulte 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 con elAWS SDK for JavaScriptpara 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 email con los resultados mediante Amazon Simple Email Service (Amazon SES).

Aprenda a:

- Cree un usuario no autenticado mediante Amazon Cognito.
- Analiza imágenes de objetos mediante Amazon Rekognition.
- Verificar una dirección de email de Amazon SES.
- Enviar una notificación por email mediante Amazon SES.

Para ver el código fuente completo y las instrucciones de configuración y ejecución, consulte el ejemplo completo en [GitHub](#).

Servicios utilizados en este ejemplo

- Amazon Rekognition
- Amazon S3
- Amazon SES

### Kotlin

#### SDK para Kotlin

##### Note

Esta es una documentación preliminar para una característica en versión de vista previa.  
Está sujeta a cambios.

Muestra cómo utilizar la API de Amazon Rekognition Kotlin 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 email con los resultados mediante Amazon Simple Email Service (Amazon SES).

Para ver el código fuente completo y las instrucciones de configuración y ejecución, consulte el ejemplo completo en [GitHub](#).

#### Servicios utilizados en este ejemplo

- Amazon Rekognition
- Amazon S3
- Amazon SES

Si desea obtener una lista completa de las guías para desarrolladores de SDK de AWS y ejemplos de código, incluida ayuda para empezar e información sobre versiones anteriores, consulte [Uso de Rekognition con un AWSSDK \(p. 15\)](#).

## Detección personas y objetos en un vídeo con Amazon Rekognition mediante un SDK de AWS

Los siguientes ejemplos de código indican cómo detectar personas y objetos en un vídeo con Amazon Rekognition.

### Java

#### SDK para Java 2.x

Muestra cómo utilizar Amazon Rekognition Java API para crear una aplicación que permita detectar rostros y objetos en vídeos ubicados en un bucket de Amazon Simple Storage Service (Amazon S3). La aplicación envía al administrador una notificación por email con los resultados mediante Amazon Simple Email Service (Amazon SES).

Para ver el código fuente completo y las instrucciones de configuración y ejecución, consulte 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 con el AWS SDK for JavaScript para crear una aplicación para detectar rostros y objetos en vídeos ubicados en un bucket de Amazon Simple Storage Service (Amazon S3). La aplicación envía al administrador una notificación por email con los resultados mediante Amazon Simple Email Service (Amazon SES).

#### Aprenda a:

- Cree un usuario no autenticado mediante Amazon Cognito.
- Analiza imágenes para EPI mediante Amazon Rekognition.

- Verificar una dirección de email de Amazon SES.
- Enviar una notificación por email mediante Amazon SES.

Para ver el código fuente completo y las instrucciones de configuración y ejecución, consulte el ejemplo completo en [GitHub](#).

#### Servicios utilizados en este ejemplo

- Amazon Rekognition
- Amazon S3
- Amazon SES

### Python

#### SDK para Python (Boto3)

Utilice Amazon Rekognition para detectar caras, objetos y personas en vídeos 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 suscribe 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 puede ver mejor en [GitHub](#). Para ver el código fuente completo y las instrucciones de configuración y ejecución, consulte el ejemplo completo en [GitHub](#).

#### Servicios utilizados en este ejemplo

- Amazon Rekognition
- Amazon SNS
- Amazon SQS

Si desea obtener una lista completa de las guías para desarrolladores de SDK de AWS y ejemplos de código, incluida ayuda para empezar e información sobre versiones anteriores, consulte [Uso de Rekognition con unAWSSDK \(p. 15\)](#).

## Guardar EXIF y otra información de imagen mediante unAWSSDK

En el siguiente ejemplo de código, se muestra cómo:

- Obtenga información EXIF de un archivo JPG, JPEG o PNG.
- Cargue el archivo de imagen en un bucket de Amazon Simple Storage Service (Amazon S3).
- Utiliza Amazon Rekognition (Amazon Rekognition) para identificar los tres atributos principales (etiquetas de Amazon Rekognition) del archivo.
- Agregue la información EXIF y etiqueta a una tabla de Amazon DynamoDB (DynamoDB) de la región.

### Rust

#### SDK para Rust

##### Note

Esta documentación es para un SDK en versión preliminar. El SDK está sujeto a cambios y no se debe utilizar en producción.

Obtenga información EXIF de un archivo JPG, JPEG o PNG, cargue el archivo de imagen en un depósito de Amazon Simple Storage Service, utilice Amazon Rekognition para identificar los tres atributos principales (labels en Amazon Rekognition) en el archivo y agregue la información EXIF y etiqueta a una tabla de Amazon DynamoDB de la región.

Para ver el código fuente completo y las instrucciones de configuración y ejecución, consulte el ejemplo completo en [GitHub](#).

#### Servicios utilizados en este ejemplo

- DynamoDB
- Amazon Rekognition
- Amazon S3

Si desea obtener una lista completa de las guías para desarrolladores de SDK de AWS y ejemplos de código, incluida ayuda para empezar e información sobre versiones anteriores, consulte [Uso de Rekognition con unAWSSDK \(p. 15\)](#).

## Ejemplos de uso de Amazon Rekognition

Los siguientes ejemplos de código muestran cómo utilizar Amazon Rekognition con AWSSDK.

### Ejemplos

- [Cree una colección de Amazon Rekognition y encuentre caras en ella mediante unAWSSDK \(p. 396\)](#)
- [Detecte y muestre elementos en imágenes con Amazon Rekognition mediante unAWSSDK \(p. 403\)](#)
- [Detecte la información de los vídeos mediante Amazon Rekognition y elAWSSDK \(p. 412\)](#)

## Cree una colección de Amazon Rekognition y encuentre caras en ella mediante unAWSSDK

El siguiente ejemplo de código muestra cómo crear una colección de Amazon Rekognition y buscar rostros en ella.

Para obtener más información, consulte [Búsqueda de rostros en una colección](#).

### Python

#### SDK para Python (Boto3)

Crea clases que envuelvan 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 envoltura para crear una colección de caras a partir de un conjunto de imágenes y, a continuación, buscar caras 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)
```

- Encuentre instrucciones y más código en [GitHub](#).

Si desea obtener una lista completa de las guías para desarrolladores de SDK de AWS y ejemplos de código, incluida ayuda para empezar e información sobre versiones anteriores, consulte [Uso de Rekognition con unAWSSDK \(p. 15\)](#).

## Detecte y muestre elementos en imágenes con Amazon Rekognition mediante unAWSSDK

El siguiente ejemplo de código muestra cómo detectar y mostrar elementos en imágenes con Amazon Rekognition.

Para obtener más información, consulte [Visualización de cuadros delimitadores](#).

Python

SDK para Python (Boto3)

Crea clases para ajustar 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 matches and %s unmatches in %s.", len(matches),
 len(unmatches), self.image_name)
 except ClientError:
 logger.exception("Couldn't compare faces in %s.", self.image_name)
 raise
 else:
 return matches, unmatches
```

```
 "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
```

Crea 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()
```

Crea clases para analizar 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.
"""\n rendering = self.face.to_dict()\n if self.name is not None:\n rendering['name'] = self.name\n if self.info_urls:\n rendering['info URLs'] = self.info_urls\n if self.timestamp is not None:\n rendering['timestamp'] = self.timestamp\n return rendering\n\n\nclass RekognitionPerson:\n """Encapsulates an Amazon Rekognition person."""\n def __init__(self, person, timestamp=None):\n """\n Initializes the person object.\n\n :param person: Person data, in the format returned by Amazon Rekognition\n functions.\n :param timestamp: The time when the person was detected, if the person\n was detected in a video.\n """\n self.index = person.get('Index')\n self.bounding_box = person.get('BoundingBox')\n face = person.get('Face')\n self.face = RekognitionFace(face) if face is not None else None\n self.timestamp = timestamp\n\n def to_dict(self):\n """\n Renders some of the person data to a dict.\n\n :return: A dict that contains the person data.\n """\n rendering = self.face.to_dict() if self.face is not None else {}\n if self.index is not None:\n rendering['index'] = self.index\n if self.bounding_box is not None:\n rendering['bounding_box'] = self.bounding_box\n if self.timestamp is not None:\n rendering['timestamp'] = self.timestamp\n return rendering\n\n\nclass RekognitionLabel:\n """Encapsulates an Amazon Rekognition label."""\n def __init__(self, label, timestamp=None):\n """\n Initializes the label object.\n\n :param label: Label data, in the format returned by Amazon Rekognition\n functions.\n :param timestamp: The time when the label was detected, if the label\n was detected in a video.\n """\n self.name = label.get('Name')\n self.confidence = label.get('Confidence')\n self.instances = label.get('Instances')\n self.parents = label.get('Parents')\n self.timestamp = timestamp\n\n def to_dict(self):\n """\n Renders some of the label data to a dict.\n\n :return: A dict that contains the label data.\n """
```

```
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 de envoltura para detectar elementos de las imágenes y mostrar sus cuadros delimitadores. Las imágenes utilizadas en este ejemplo se pueden encontrar en GitHub junto con 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)
```

- Encuentre instrucciones y más código en [GitHub](#).

Si desea obtener una lista completa de las guías para desarrolladores de SDK de AWS y ejemplos de código, incluida ayuda para empezar e información sobre versiones anteriores, consulte [Uso de Rekognition con unAWSSDK \(p. 15\)](#).

## Detecte la información de los vídeos mediante Amazon Rekognition y el AWSSDK

Los siguientes ejemplos de código muestran cómo detectar información en los vídeos.

Java

### SDK para Java 2.x

Obtenga resultados de celebridades en un vídeo ubicado en un bucket de Amazon S3.

```
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: " + Long.toString(seconds) + " ");
 CelebrityDetail details=celeb.celebrity();
 System.out.println("Name: " + details.name());
 System.out.println("Id: " + details.id());
 System.out.println();
}

} while (recognitionResponse !=null &&
recognitionResponse.nextToken() != null);

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

Detección etiquetas en un vídeo mediante una operación de detección de etiquetas.

```
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=null;
 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();
} catch (Exception 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: " + Long.toString(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);
}
}
```

Detección rostros en un vídeo almacenado en un bucket de Amazon S3.

```
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=null;
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();
} catch (Exception 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: " + Long.toString(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);
}
}
```

Detectar contenido inapropiado u ofensivo en un vídeo almacenado en un bucket de Amazon S3.

```
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);
}
}
```

Detección segmentos de indicaciones técnicas y segmentos de detección de tomas en un vídeo almacenado en un bucket de Amazon S3.

```
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> detectedSegment =
segDetectionResponse.segments();
 String type = detectedSegment.get(0).type().toString();

 if (type.contains(SegmentType.TECHNICAL_CUE.toString())) {
 System.out.println("Technical Cue");
 TechnicalCueSegment segmentCue =
detectedSegment.get(0).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.get(0).shotSegment();
 System.out.println("\tIndex " + segmentShot.index());
 System.out.println("\tConfidence: " +
segmentShot.confidence().toString());
 }
 long seconds = detectedSegment.get(0).durationMillis();
 System.out.println("\tDuration : " + Long.toString(seconds) + " "
milliseconds);
 System.out.println("\tStart time code: " +
detectedSegment.get(0).startTimecodeSMPTE());
 System.out.println("\tEnd time code: " +
detectedSegment.get(0).endTimecodeSMPTE());
 System.out.println("\tDuration time code: " +
detectedSegment.get(0).durationSMPTE());
 System.out.println();

} while (segDetectionResponse !=null && segDetectionResponse.nextToken() != null);

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

Detección texto en un vídeo almacenado en un vídeo almacenado en un bucket de Amazon S3.

```
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);
}
}
```

Detección personas en un vídeo almacenado en un vídeo almacenado en un bucket de Amazon S3.

```
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);
}
}
```

- Encuentre instrucciones y más código en [GitHub](#).

## Kotlin

### SDK para Kotlin

#### Note

Esta es una documentación preliminar para una característica en versión de vista previa.  
Está sujeta a cambios.

Detección 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 { 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 { 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}")
 }
 }
}
```

Detectar 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 { 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)
 }
 }
}
```

- Encuentre instrucciones y más código en [GitHub](#).

Si desea obtener una lista completa de las guías para desarrolladores de SDK de AWS y ejemplos de código, incluida ayuda para empezar e información sobre versiones anteriores, consulte [Uso de Rekognition con unAWSSDK \(p. 15\)](#).

# Ejemplos de API para Amazon Rekognition

Los siguientes ejemplos de código muestran cómo utilizar Amazon Rekognition con AWSSDK.

## Ejemplos

- [Comparar los rostros de una imagen con una imagen de referencia con Amazon Rekognition mediante unAWSSDK \(p. 430\)](#)
- [Crear una colección de Amazon Rekognition utilizando unAWSSDK \(p. 435\)](#)
- [Eliminar una colección de Amazon Rekognition mediante unAWSSDK \(p. 437\)](#)
- [Eliminar caras de una colección de Amazon Rekognition mediante unAWSSDK \(p. 439\)](#)
- [Describir una colección de Amazon Rekognition mediante unAWSSDK \(p. 442\)](#)
- [Detectar rostros de una imagen con Amazon Rekognition mediante unAWSSDK \(p. 445\)](#)
- [Detectar etiquetas de una imagen con Amazon Rekognition mediante unAWSSDK \(p. 450\)](#)
- [Detección etiquetas de moderación en una imagen con Amazon Rekognition mediante unAWSSDK \(p. 454\)](#)
- [Detectar texto de una imagen con Amazon Rekognition mediante unAWSSDK \(p. 457\)](#)
- [Obtenga información sobre celebridades con Amazon Rekognition utilizando unAWSSDK \(p. 460\)](#)
- [Indexar caras de una colección de Amazon Rekognition mediante unAWSSDK \(p. 461\)](#)
- [Enumerar colecciones de Amazon Rekognition utilizando unAWSSDK \(p. 465\)](#)
- [Enumerar caras de una colección de Amazon Rekognition utilizando unAWSSDK \(p. 468\)](#)
- [Reconoce a las celebridades de una imagen con Amazon Rekognition usando unAWSSDK \(p. 471\)](#)
- [Buscar rostros en una colección de Amazon Rekognition mediante unAWSSDK \(p. 474\)](#)
- [Buscar rostros en una colección de Amazon Rekognition en comparación con una imagen de referencia mediante unAWSSDK \(p. 477\)](#)

## Comparar los rostros de una imagen con una imagen de referencia con Amazon Rekognition mediante unAWSSDK

Los siguientes ejemplos de código muestran cómo comparar rostros en una imagen con una imagen de referencia con Amazon Rekognition.

Para obtener más información, consulte [Comparación de rostros en imágenes](#).

### .NET

#### AWS SDK for .NET

```
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.");
}
```

- Encuentre instrucciones y más código en [GitHub](#).
- Para obtener más información, consulte [CompareFaces](#)en AWS SDK for .NET Referencia de la API.

---

Java

SDK para Java 2.x

```
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> uncompered = compareFacesResult.unmatchedFaces();
 System.out.println("There was " + uncompered.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);
 }
}
```

- Encuentre instrucciones y más código en [GitHub](#).
- Para obtener más información, consulte [CompareFaces](#) en AWS SDK for Java 2.x Reference de la API.

---

Kotlin

SDK para Kotlin

Note

Esta es una documentación preliminar para una característica en versión de vista previa.  
Está sujeta a cambios.

```
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 { 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.toString()}
${position.top} matches with ${face.confidence.toString()} % confidence.")
 }
 }

 val uncompered = compareFacesResult.unmatchedFaces
 if (uncompered != null)
 println("There was ${uncompered.size} face(s) that did not match")

 println("Source image rotation:
${compareFacesResult.sourceImageOrientationCorrection}")
 println("target image rotation:
${compareFacesResult.targetImageOrientationCorrection}")
 }
}
```

- Encuentre instrucciones y más código en [GitHub](#).
- Para obtener más información, consulte [CompareFaces](#)en AWSSDK para la API de Kotlin.

Python

SDK para Python (Boto3)

```
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
```

- Encuentre instrucciones y más código en [GitHub](#).
- Para obtener más información, consulte [CompareFaces](#)en AWS Referencia de API SDK for Python (Boto3).

Si desea obtener una lista completa de las guías para desarrolladores de SDK de AWS y ejemplos de código, incluida ayuda para empezar e información sobre versiones anteriores, consulte [Uso de Rekognition con un AWSSDK \(p. 15\)](#).

## Crear una colección de Amazon Rekognition utilizando unAWSSDK

Los siguientes ejemplos de código muestran cómo crear una colección de Amazon Rekognition.

Para obtener más información, consulte[Creación de una colección](#).

.NET

AWS SDK for .NET

```
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}");
}
```

- Encuentre instrucciones y más código en [GitHub](#).
- Para obtener más información, consulte[CreateCollection](#)enAWS SDK for .NETReferencia de la API.

Java

SDK para Java 2.x

```
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);
 }
}
```

- Encuentre instrucciones y más código en [GitHub](#).
- Para obtener más información, consulte [CreateCollection](#)en AWS SDK for Java 2.xReferencia de la API.

## Kotlin

### SDK para Kotlin

#### Note

Esta es una documentación preliminar para una característica en versión de vista previa.  
Está sujeta a cambios.

```
suspend fun createMyCollection(collectionIdVal: String) {

 val request = CreateCollectionRequest {
 collectionId = collectionIdVal
 }

 RekognitionClient { region = "us-east-1" }.use { rekClient ->
 val response = rekClient.createCollection(request)
 println("Collection ARN is ${response.collectionArn}")
 println("Status code is ${response.statusCode} ")
 }
}
```

- Encuentre instrucciones y más código en [GitHub](#).
- Para obtener más información, consulte [CreateCollection](#)en AWSSDK para la API de Kotlin.

## Python

### SDK para Python (Boto3)

```
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
```

- Encuentre instrucciones y más código en [GitHub](#).
- Para obtener más información, consulte [CreateCollection](#)enAWSReferencia de API SDK for Python (Boto3).

Si desea obtener una lista completa de las guías para desarrolladores de SDK de AWS y ejemplos de código, incluida ayuda para empezar e información sobre versiones anteriores, consulte [Uso de Rekognition con unAWSSDK \(p. 15\)](#).

## Eliminar una colección de Amazon Rekognition mediante unAWSSDK

Los siguientes ejemplos de código muestran cómo eliminar una colección de Amazon Rekognition.

Para obtener más información, consulte [Eliminación de una colección](#).

.NET

AWS SDK for .NET

```
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}");
}
```

- Encuentre instrucciones y más código en [GitHub](#).
- Para obtener más información, consulte [DeleteCollection](#)enAWS SDK for .NETReferencia de la API.

## Java

### SDK para Java 2.x

```
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);
 }
}
```

- Encuentre instrucciones y más código en [GitHub](#).
- Para obtener más información, consulte [DeleteCollection](#)enAWS SDK for Java 2.xReferencia de la API.

## Kotlin

### SDK para Kotlin

#### Note

Esta es una documentación preliminar para una característica en versión de vista previa.  
Está sujeta a cambios.

```
suspend fun deleteMyCollection(collectionIdVal: String) {

 val request = DeleteCollectionRequest {
 collectionId = collectionIdVal
 }

 RekognitionClient { region = "us-east-1" }.use { rekClient ->
 val response = rekClient.deleteCollection(request)
 println("The collectionId status is ${response.statusCode.toString()}")
 }
}
```

- Encuentre instrucciones y más código en [GitHub](#).
- Para obtener más información, consulte [DeleteCollection](#)enAWSSDK para la API de Kotlin.

## Python

### SDK para Python (Boto3)

```
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
```

- Encuentre instrucciones y más código en [GitHub](#).
- Para obtener más información, consulte [DeleteCollectionenAWSReferencia de API SDK for Python \(Boto3\)](#).

Si desea obtener una lista completa de las guías para desarrolladores de SDK de AWS y ejemplos de código, incluida ayuda para empezar e información sobre versiones anteriores, consulte [Uso de Rekognition con unAWSSDK \(p. 15\)](#).

## Eliminar caras de una colección de Amazon Rekognition mediante unAWSSDK

Los siguientes ejemplos de código muestran cómo eliminar rostros de una colección de Amazon Rekognition.

Para obtener más información, consulte[Eliminación de rostros de una colección](#).

#### .NET

##### AWS SDK for .NET

```
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}");
 });
}
```

- Encuentre instrucciones y más código en [GitHub](#).
- Para obtener más información, consulte[DeleteFacesenAWS SDK for .NETReferencia de la API](#).

#### Java

##### SDK para Java 2.x

```
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);
 }
}
```

- Encuentre instrucciones y más código en [GitHub](#).

- Para obtener más información, consulte [DeleteFacesenAWS SDK for Java 2.xReferencia de la API](#).

## Kotlin

### SDK para Kotlin

#### Note

Esta es una documentación preliminar para una característica en versión de vista previa.  
Está sujeta a cambios.

```
suspend fun deleteFacesCollection(collectionIdVal: String?, faceIdVal: String) {

 val deleteFacesRequest = DeleteFacesRequest {
 collectionId = collectionIdVal
 faceIds = listOf(faceIdVal)
 }

 RekognitionClient { region = "us-east-1" }.use { rekClient ->
 rekClient.deleteFaces(deleteFacesRequest)
 println("$faceIdVal was deleted from the collection")
 }
}
```

- Encuentre instrucciones y más código en [GitHub](#).
- Para obtener más información, consulte [DeleteFacesenAWSSDK para la API de Kotlin](#).

## Python

### SDK para Python (Boto3)

```
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
```

- Encuentre instrucciones y más código en [GitHub](#).
- Para obtener más información, consulte [DeleteFacesenAWSReferencia de API SDK for Python \(Boto3\)](#).

Si desea obtener una lista completa de las guías para desarrolladores de SDK de AWS y ejemplos de código, incluida ayuda para empezar e información sobre versiones anteriores, consulte [Uso de Rekognition con unAWSSDK \(p. 15\)](#).

## Describir una colección de Amazon Rekognition mediante unAWSSDK

Los siguientes ejemplos de código muestran cómo describir una colección de Amazon Rekognition.

Para obtener más información, consulte [Descripción de una colección](#).

.NET

AWS SDK for .NET

```
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}");
}
```

- Encuentre instrucciones y más código en [GitHub](#).
- Para obtener más información, consulte [DescribeCollection](#)enAWS SDK for .NETReferencia de la API.

## Java

### SDK para Java 2.x

```
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);
 }
}
```

- Encuentre instrucciones y más código en [GitHub](#).
- Para obtener más información, consulte [DescribeCollection](#)enAWS SDK for Java 2.xReferencia de la API.

## Kotlin

### SDK para Kotlin

#### Note

Esta es una documentación preliminar para una característica en versión de vista previa.  
Está sujeta a cambios.

```
suspend fun describeColl(collectionName: String) {
```

```
 val request = DescribeCollectionRequest {
 collectionId = collectionName
 }

 RekognitionClient { 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}")
 }
}
```

- Encuentre instrucciones y más código en [GitHub](#).
- Para obtener más información, consulte [DescribeCollection](#)en AWSSDK para la API de Kotlin.

## Python

### SDK para Python (Boto3)

```
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()
```

- Encuentre instrucciones y más código en [GitHub](#).
- Para obtener más información, consulte [DescribeCollection](#)en AWS Referencia de API SDK for Python (Boto3).

Si desea obtener una lista completa de las guías para desarrolladores de SDK de AWS y ejemplos de código, incluida ayuda para empezar e información sobre versiones anteriores, consulte [Uso de Rekognition con unAWSSDK \(p. 15\)](#).

## Detectar rostros de una imagen con Amazon Rekognition mediante unAWSSDK

Los siguientes ejemplos de código muestran cómo detectar rostros en una imagen con Amazon Rekognition.

Para obtener más información, consulte [Detección de rostros en una imagen](#).

.NET

AWS SDK for .NET

```
public static async Task Main()
{
 string photo = "input.jpg";
 string bucket = "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);
 }
}
```

Mostrar información del cuadro delimitador de todas las caras de una imagen.

```
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}");
}
```

- Encuentre instrucciones y más código en [GitHub](#).
- Para obtener más información, consulte [DetectFaces en AWS SDK for .NET](#) Referencia de la API.

## Java

### SDK para Java 2.x

```
public static void detectFacesinImage(RekognitionClient rekClient, String
sourceImage) {

 try {
 InputStream sourceStream = new FileInputStream(new File(sourceImage));
 SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);

 // Create an Image object for the source image.
 Image souImage = Image.builder()
 .bytes(sourceBytes)
 .build();

 DetectFacesRequest facesRequest = DetectFacesRequest.builder()
 .attributes(Attribute.ALL)
 .image(souImage)
 .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 | FileNotFoundException e) {
 System.out.println(e.getMessage());
 System.exit(1);
}
}
```

- Encuentre instrucciones y más código en [GitHub](#).
- Para obtener más información, consulte [DetectFacesenAWS SDK for Java 2.xReferencia de la API](#).

## Kotlin

### SDK para Kotlin

#### Note

Esta es una documentación preliminar para una característica en versión de vista previa.  
Está sujeta a cambios.

```
suspend fun detectFacesinImage(sourceImage: String?) {

 val souImage = Image {
 bytes = (File(sourceImage).readBytes())
 }

 val request = DetectFacesRequest {
 attributes = listOf(Attribute.All)
 image = souImage
 }

 RekognitionClient { 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.toString()} and ${ageRange?.high.toString()} years old.")
 println("There is a smile ${face.smile?.value.toString()}")
 }
 }
}
```

- Encuentre instrucciones y más código en [GitHub](#).
- Para obtener más información, consulte [DetectFacesenAWSSDK para la API de Kotlin](#).

## Python

### SDK para Python (Boto3)

```
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
```

- Encuentre instrucciones y más código en [GitHub](#).
- Para obtener más información, consulte [DetectFacesenAWSReferencia de API SDK for Python \(Boto3\)](#).

Si desea obtener una lista completa de las guías para desarrolladores de SDK de AWS y ejemplos de código, incluida ayuda para empezar e información sobre versiones anteriores, consulte [Uso de Rekognition con unAWSSDK \(p. 15\)](#).

## Detectar etiquetas de una imagen con Amazon Rekognition mediante unAWSSDK

Los siguientes ejemplos de código muestran cómo detectar etiquetas en una imagen con Amazon Rekognition.

Para obtener más información, consulte [Detección de etiquetas en una imagen](#).

.NET

AWS SDK for .NET

```
public static async Task Main()
{
 string photo = "del_river_02092020_01.jpg"; // "input.jpg";
```

```
string bucket = "igsmiths3photos"; // "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 etiquetas en un archivo de imagen almacenado en el equipo.

```
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);
 }
}
```

- Encuentre instrucciones y más código en [GitHub](#).
- Para obtener más información, consulte [DetectLabels en AWS SDK for .NET Referencia de la API](#).

## Java

### SDK para Java 2.x

```
public static void detectImageLabels(RekognitionClient rekClient, String
sourceImage) {

 try {

 InputStream sourceStream = new URL("https://images.unsplash.com/
photo-1557456170-0cf4f4d0d362?ixid=MnwxMjA3fDB8MHxzZWFyY2h8MXx8bGFrZXlb
nwwfHwwfHw
%3D&ixlib=rb-1.2.1&w=1000&q=80").openStream();
 // InputStream sourceStream = new FileInputStream(sourceImage);
 SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);

 // 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);
 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 | FileNotFoundException |
MalformedURLException e) {
```

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

- Encuentre instrucciones y más código en [GitHub](#).
- Para obtener más información, consulte [DetectLabelsenAWS SDK for Java 2.xReferencia de la API](#).

## Kotlin

### SDK para Kotlin

#### Note

Esta es una documentación preliminar para una característica en versión de vista previa.  
Está sujeta a cambios.

```
suspend fun detectImageLabels(sourceImage: String) {

 val souImage = Image {
 bytes = (File(sourceImage).readBytes())
 }
 val request = DetectLabelsRequest {
 image = souImage
 maxLabels = 10
 }

 RekognitionClient { region = "us-east-1" }.use { rekClient ->
 val response = rekClient.detectLabels(request)
 response.labels?.forEach { label ->
 println("${label.name} : ${label.confidence}")
 }
 }
}
```

- Encuentre instrucciones y más código en [GitHub](#).
- Para obtener más información, consulte [DetectLabelsenAWSSDK para la API de Kotlin](#).

## Python

### SDK para Python (Boto3)

```
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
```

- Encuentre instrucciones y más código en [GitHub](#).
- Para obtener más información, consulte [DetectLabels](#)en AWS Referencia de API SDK for Python (Boto3).

Si desea obtener una lista completa de las guías para desarrolladores de SDK de AWS y ejemplos de código, incluida ayuda para empezar e información sobre versiones anteriores, consulte [Uso de Rekognition con unAWSSDK](#) (p. 15).

## Detección etiquetas de moderación en una imagen con Amazon Rekognition mediante unAWSSDK

Los siguientes ejemplos de código muestran cómo detectar etiquetas de moderación en una imagen con Amazon Rekognition. Las etiquetas de moderación identifican el contenido que puede ser inapropiado para algunas audiencias.

Para obtener más información, consulte [Detección de imágenes inapropiadas](#).

.NET

AWS SDK for .NET

```
public static async Task Main(string[] args)
{
 string photo = "input.jpg";
 string bucket = "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);
 }
 }
}
```

- Encuentre instrucciones y más código en [GitHub](#).
- Para obtener más información, consulte [DetectModerationLabels](#)en AWS SDK for .NETReferencia de la API.

## Java

### SDK para Java 2.x

```
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);

 // Display the results
 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);
 }
}
```

- Encuentre instrucciones y más código en [GitHub](#).
- Para obtener más información, consulte [DetectModerationLabels](#)enAWS SDK for Java 2.xReferencia de la API.

## Kotlin

### SDK para Kotlin

#### Note

Esta es una documentación preliminar para una característica en versión de vista previa.  
Está sujeta a cambios.

```
suspend fun detectModLabels(sourceImage: String) {

 val myImage = Image {
 this.bytes = (File(sourceImage).readBytes())
 }

 val request = DetectModerationLabelsRequest {
 image = myImage
 minConfidence = 60f
 }

 RekognitionClient { 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}")
 }
 }
}
```

- Encuentre instrucciones y más código en [GitHub](#).
- Para obtener más información, consulte [DetectModerationLabels](#)enAWSSDK para la API de Kotlin.

## Python

### SDK para Python (Boto3)

```
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
```

- Encuentre instrucciones y más código en [GitHub](#).
- Para obtener más información, consulte [DetectModerationLabels](#) en [AWS Referencia de API SDK for Python \(Boto3\)](#).

Si desea obtener una lista completa de las guías para desarrolladores de SDK de AWS y ejemplos de código, incluida ayuda para empezar e información sobre versiones anteriores, consulte [Uso de Rekognition con unAWSSDK \(p. 15\)](#).

## Detectar texto de una imagen con Amazon Rekognition mediante unAWSSDK

Los siguientes ejemplos de código muestran cómo detectar texto en una imagen con Amazon Rekognition.

Para obtener más información, consulte [Detección de texto en una imagen](#).

.NET

AWS SDK for .NET

```
public static async Task Main()
```

```
{
 string photo = "Dad_photographer.jpg"; // "input.jpg";
 string bucket = "igsmiths3photos"; // "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);
 }
}
```

- Encuentre instrucciones y más código en [GitHub](#).
- Para obtener más información, consulte [DetectTextenAWS SDK for .NETReferencia de la API](#).

## Java

### SDK para Java 2.x

```
public static void detectTextLabels(RekognitionClient rekClient, String
sourceImage) {

 try {

 InputStream sourceStream = new FileInputStream(sourceImage);
 SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);

 // Create an Image object for the source image
 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);
}
}
```

- Encuentre instrucciones y más código en [GitHub](#).
- Para obtener más información, consulte [DetectTextenAWS SDK for Java 2.xReferencia de la API](#).

## Kotlin

### SDK para Kotlin

#### Note

Esta es una documentación preliminar para una característica en versión de vista previa.  
Está sujeta a cambios.

```
suspend fun detectTextLabels(sourceImage: String?) {

 val souImage = Image {
 bytes = (File(sourceImage).readBytes())
 }

 val request = DetectTextRequest {
 image = souImage
 }

 RekognitionClient { 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}")
 }
 }
}
```

- Encuentre instrucciones y más código en [GitHub](#).
- Para obtener más información, consulte [DetectTextenAWSSDK para la API de Kotlin](#).

## Python

### SDK para Python (Boto3)

```
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
```

- Encuentre instrucciones y más código en [GitHub](#).
- Para obtener más información, consulte [DetectText](#)en AWS Referencia de API SDK for Python (Boto3).

Si desea obtener una lista completa de las guías para desarrolladores de SDK de AWS y ejemplos de código, incluida ayuda para empezar e información sobre versiones anteriores, consulte [Uso de Rekognition con un AWSSDK](#) (p. 15).

## Obtenga información sobre celebridades con Amazon Rekognition utilizando un AWSSDK

El siguiente ejemplo de código muestra cómo obtener información sobre celebridades mediante Amazon Rekognition.

### .NET

#### AWS SDK for .NET

```
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):");
 celebrityInfoResponseUrls.ForEach(url =>
 {
 Console.WriteLine(url);
 });
}
```

- Encuentre instrucciones y más código en [GitHub](#).
- Para obtener más información, consulte [GetCelebrityInfo en AWS SDK for .NET](#) Referencia de la API.

Si desea obtener una lista completa de las guías para desarrolladores de SDK de AWS y ejemplos de código, incluida ayuda para empezar e información sobre versiones anteriores, consulte [Uso de Rekognition con unAWSSDK \(p. 15\)](#).

## Indexar caras de una colección de Amazon Rekognition mediante unAWSSDK

Los siguientes ejemplos de código muestran cómo indexar rostros en una imagen y añadirlas a una colección de Amazon Rekognition.

Para obtener más información, consulte [Adición de rostros a una colección](#).

.NET

AWS SDK for .NET

```
public static async Task Main()
{
 string collectionId = "MyCollection2";
 string bucket = "doc-example-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}");
}
}
```

- Encuentre instrucciones y más código en [GitHub](#).
- Para obtener más información, consulte [IndexFaces en AWS SDK for .NET](#) Referencia de la API.

## Java

### SDK para Java 2.x

```
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);

 // Display the results.
 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);
}
}
```

- Encuentre instrucciones y más código en [GitHub](#).
- Para obtener más información, consulte [IndexFacesenAWS SDK for Java 2.xReferencia de la API](#).

## Kotlin

### SDK para Kotlin

#### Note

Esta es una documentación preliminar para una característica en versión de vista previa.  
Está sujeta a cambios.

```
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 { 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.toString()}")
 }

 println("Faces not indexed:")
 facesResponse.unindexedFaces?.forEach { unindexedFace ->
```

```
 println("Location:
${unindexedFace.faceDetail?.boundingBox.toString()}")
 println("Reasons:")

 unindexedFace.reasons?.forEach { reason ->
 println("Reason: $reason")
 }
 }
}
```

- Encuentre instrucciones y más código en [GitHub](#).
- Para obtener más información, consulte [IndexFacesenAWSSDK para la API de Kotlin](#).

## Python

### SDK para Python (Boto3)

```
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
```

- Encuentre instrucciones y más código en [GitHub](#).
- Para obtener más información, consulte [IndexFaces en AWS Reference de API SDK for Python \(Boto3\)](#).

Si desea obtener una lista completa de las guías para desarrolladores de SDK de AWS y ejemplos de código, incluida ayuda para empezar e información sobre versiones anteriores, consulte [Uso de Rekognition con unAWSSDK \(p. 15\)](#).

## Enumerar colecciones de Amazon Rekognition utilizando unAWSSDK

Los siguientes ejemplos de código muestran cómo enumerar las colecciones de Amazon Rekognition.

Para obtener más información, consulte [Listado de colecciones](#).

.NET

### AWS SDK for .NET

```
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);
}
```

- Encuentre instrucciones y más código en [GitHub](#).
- Para obtener más información, consulte [ListCollections](#)enAWS SDK for .NETReferencia de la API.

## Java

### SDK para Java 2.x

```
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);
 }
}
```

- Encuentre instrucciones y más código en [GitHub](#).
- Para obtener más información, consulte [ListCollections](#)enAWS SDK for Java 2.xReferencia de la API.

## Kotlin

### SDK para Kotlin

#### Note

Esta es una documentación preliminar para una característica en versión de vista previa.  
Está sujeta a cambios.

```
suspend fun listAllCollections() {

 val request = ListCollectionsRequest {
 maxResults = 10
 }

 RekognitionClient { region = "us-east-1" }.use { rekClient ->
 val response = rekClient.listCollections(request)
 response.collectionIds?.forEach { resultId ->
 println(resultId)
 }
 }
}
```

- Encuentre instrucciones y más código en [GitHub](#).
- Para obtener más información, consulte [ListCollections](#)en AWS SDK para la API de Kotlin.

#### Python

##### SDK para Python (Boto3)

```
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
```

- Encuentre instrucciones y más código en [GitHub](#).
- Para obtener más información, consulte [ListCollections](#)en AWS Referencia de API SDK for Python (Boto3).

Si desea obtener una lista completa de las guías para desarrolladores de SDK de AWS y ejemplos de código, incluida ayuda para empezar e información sobre versiones anteriores, consulte [Uso de Rekognition con unAWSSDK \(p. 15\)](#).

## Enumerar caras de una colección de Amazon Rekognition utilizando unAWSSDK

Los siguientes ejemplos de código muestran cómo enumerar rostros en una colección de Amazon Rekognition.

Para obtener más información, consulte [Mostrar una lista de rostros en una colección](#).

.NET

AWS SDK for .NET

```
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));
}
```

- Encuentre instrucciones y más código en [GitHub](#).
- Para obtener más información, consulte [ListFaces](#)enAWS SDK for .NETReferencia de la API.

Java

SDK para Java 2.x

```
public static void listFacesCollection(RekognitionClient rekClient, String
collectionId) {

 try {
```

```
ListFacesRequest facesRequest = ListFacesRequest.builder()
 .collectionId(collectionId)
 .maxResults(10)
 .build();

ListFacesResponse facesResponse = rekClient.listFaces(facesRequest);

// For each face in the collection, print out the confidence level and
face id value.
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);
}
}
```

- Encuentre instrucciones y más código en [GitHub](#).
- Para obtener más información, consulte [ListFaces](#)enAWS SDK for Java 2.xReferencia de la API.

## Kotlin

### SDK para Kotlin

#### Note

Esta es una documentación preliminar para una característica en versión de vista previa.  
Está sujeta a cambios.

```
suspend fun listFacesCollection(collectionIdVal: String?) {

 val request = ListFacesRequest {
 collectionId = collectionIdVal
 maxResults = 10
 }

 RekognitionClient { 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()}")
 }
 }
}
```

- Encuentre instrucciones y más código en [GitHub](#).
- Para obtener más información, consulte [ListFaces](#)enAWSSDK para la API de Kotlin.

## Python

### SDK para Python (Boto3)

```
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
```

- Encuentre instrucciones y más código en [GitHub](#).
- Para obtener más información sobre API, consulte [ListFaces](#)en AWS Referencia de API de SDK for Python (Boto3).

Si desea obtener una lista completa de las guías para desarrolladores de SDK de AWS y ejemplos de código, incluida ayuda para empezar e información sobre versiones anteriores, consulte [Uso de Rekognition con un AWSSDK \(p. 15\)](#).

## Reconoce a las celebridades de una imagen con Amazon Rekognition usando un AWSSDK

Los siguientes ejemplos de código muestran cómo reconocer celebridades en una imagen con Amazon Rekognition.

Para obtener más información, consulte [Reconocimiento de famosos en una imagen](#).

.NET

AWS SDK for .NET

```
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):");
 celebUrls.ForEach(url =>
 {
 Console.WriteLine(url);
 });
});

Console.WriteLine($"{recognizeCelebritiesResponse.UnrecognizedFaces.Count} face(s)
were unrecognized.");
}
```

```
}
```

- Encuentre instrucciones y más código en [GitHub](#).
- Para obtener más información, consulte [RecognizeCelebrities](#)enAWS SDK for .NETReferencia de la API.

## Java

### SDK para Java 2.x

```
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);
 }
}
```

- Encuentre instrucciones y más código en [GitHub](#).
- Para obtener más información sobre API, consulte [RecognizeCelebrities](#)enAWS SDK for Java 2.xReferencia de la API.

## Kotlin

### SDK para Kotlin

#### Note

Esta es una documentación preliminar para una característica en versión de vista previa.  
Está sujeta a cambios.

```
suspend fun recognizeAllCelebrities(sourceImage: String?) {

 val souImage = Image {
 bytes = (File(sourceImage).readBytes())
 }

 val request = RecognizeCelebritiesRequest{
 image = souImage
 }

 RekognitionClient { 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.")
 }
}
```

- Encuentre instrucciones y más código en [GitHub](#).
- Para obtener más información sobre API, consulte [RecognizeCelebrities](#)en AWSSDK para la API de Kotlin.

## Python

### SDK para Python (Boto3)

```
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
```

- Encuentre instrucciones y más código en [GitHub](#).
- Para obtener más información sobre API, consulte [RecognizeCelebrities](#) en AWS Referencia de API de SDK for Python (Boto3).

Si desea obtener una lista completa de las guías para desarrolladores de SDK de AWS y ejemplos de código, incluida ayuda para empezar e información sobre versiones anteriores, consulte [Uso de Rekognition con unAWSSDK \(p. 15\)](#).

## Buscar rostros en una colección de Amazon Rekognition mediante unAWSSDK

En los siguientes ejemplos de código se muestra cómo buscar caras en una colección de Amazon Rekognition que coinciden con otra cara de la colección.

Para obtener más información, consulte [Búsqueda de un rostro \(ID de cara\)](#).

.NET

AWS SDK for .NET

```
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}");
});
}
```

- Encuentre instrucciones y más código en [GitHub](#).
- Para obtener más información sobre API, consulte [SearchFaces en AWS SDK for .NET](#) Referencia de la API.

## Java

### SDK para Java 2.x

```
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) ;

 // Display the results.
 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);
 }
}
```

- Encuentre instrucciones y más código en [GitHub](#).

- Para obtener más información sobre API, consulte [SearchFaces](#)en AWS SDK for Java 2.xReferencia de la API.

## Python

### SDK para Python (Boto3)

```
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
```

- Encuentre instrucciones y más código en [GitHub](#).
- Para obtener más información sobre API, consulte [SearchFacesenAWSReferencia de API de SDK for Python \(Boto3\)](#).

Si desea obtener una lista completa de las guías para desarrolladores de SDK de AWS y ejemplos de código, incluida ayuda para empezar e información sobre versiones anteriores, consulte [Uso de Rekognition con unAWSSDK \(p. 15\)](#).

## Buscar rostros en una colección de Amazon Rekognition en comparación con una imagen de referencia mediante unAWSSDK

Los siguientes ejemplos de código indican cómo buscar caras en una colección de Amazon Rekognition en comparación con una imagen de referencia.

Para obtener más información, consulte [Búsqueda de un rostro \(imagen\)](#).

.NET

AWS SDK for .NET

```
public static async Task Main()
{
 string collectionId = "MyCollection";
 string bucket = "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}");
});
}
```

- Encuentre instrucciones y más código en [GitHub](#).
- Para obtener más información sobre API, consulte [SearchFacesByImageenAWS SDK for .NETReferencia de la API](#).

## Java

### SDK para Java 2.x

```
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) ;

 // Display the results.
 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);
 }
}
```

- Encuentre instrucciones y más código en [GitHub](#).
- Para obtener más información sobre API, consulte [SearchFacesByImageenAWS SDK for Java 2.xReferencia de la API](#).

## Python

### SDK para Python (Boto3)

```
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
```

- Encuentre instrucciones y más código en [GitHub](#).
- Para obtener más información sobre API, consulte [SearchFacesByImageenAWSReferencia de API de SDK for Python \(Boto3\)](#).

Si desea obtener una lista completa de las guías para desarrolladores de SDK de AWS y ejemplos de código, incluida ayuda para empezar e información sobre versiones anteriores, consulte [Uso de Rekognition con unAWSSDK \(p. 15\)](#).

# Seguridad de Amazon Rekognition

La seguridad en la nube de AWS es la mayor prioridad. Como cliente de AWS, se beneficiará de una arquitectura de red y un centro de datos que están diseñados para satisfacer los requisitos de seguridad de las organizaciones más exigentes.

Utilice los siguientes temas para aprender a proteger los recursos de Amazon Rekognition.

## Temas

- [Identity and Access Management para Amazon Rekognition \(p. 481\)](#)
- [Protección de datos en Amazon Rekognition \(p. 498\)](#)
- [Monitorización Rekognition \(p. 500\)](#)
- [Registro de llamadas a la API de Amazon Rekognition con AWS CloudTrail \(p. 504\)](#)
- [Uso de Amazon Rekognition con endpoints de Amazon VPC \(p. 507\)](#)
- [Validación de la conformidad para Amazon Rekognition \(p. 509\)](#)
- [Resiliencia en Amazon Rekognition \(p. 510\)](#)
- [Configuración y análisis de vulnerabilidades en Amazon Rekognition \(p. 510\)](#)
- [Seguridad de la infraestructura en Amazon Rekognition \(p. 510\)](#)

## Identity and Access Management para Amazon Rekognition

AWS Identity and Access Management (IAM) es un servicio de AWS que ayuda al administrador a controlar de forma segura el acceso a los recursos de AWS. Los administradores de IAM controlan quién puede ser autenticado (iniciado sesión) y autorizado (tienen permisos) para utilizar los recursos de Amazon Rekognition. IAM es un servicio de AWS que puede utilizar sin cargo adicional.

## Temas

- [Público \(p. 481\)](#)
- [Autenticación con identidades \(p. 482\)](#)
- [Administración de acceso mediante políticas \(p. 484\)](#)
- [Cómo funciona Amazon Rekognition con IAM \(p. 486\)](#)
- [AWS Políticas administradas por para Amazon Rekognition \(p. 489\)](#)
- [Ejemplos de políticas de Amazon Rekognition basadas en identidades \(p. 493\)](#)
- [Solución de problemas de identidad y acceso de Amazon Rekognition \(p. 496\)](#)

## Público

Cómo se utiliza AWS Identity and Access Management (IAM) varía, en función del trabajo que realice en Amazon Rekognition.

Usuario de servicio: si utiliza el servicio Amazon Rekognition para realizar su trabajo, su administrador le proporciona las credenciales y los permisos que necesita. A medida que utilice más características de Amazon Rekognition para realizar su trabajo, es posible que necesite permisos adicionales. Entender cómo se administra el acceso puede ayudarle a solicitar los permisos correctos a su administrador. Si no

puede acceder a una característica de Amazon Rekognition, consulte [Solución de problemas de identidad y acceso de Amazon Rekognition \(p. 496\)](#).

**Administrador de servicios**— Si está a cargo de los recursos de Amazon Rekognition de su empresa, probablemente disponga de acceso completo a Amazon Rekognition. Su trabajo consiste en determinar a qué características y recursos de Amazon Rekognition pueden acceder los empleados. A continuación, debe enviar solicitudes a su administrador de IAM para cambiar los permisos de los usuarios de su servicio. Revise la información de esta página para conocer los conceptos básicos de IAM. Para obtener más información sobre cómo su empresa puede utilizar IAM con Amazon Rekognition, consulte [Cómo funciona Amazon Rekognition con IAM \(p. 486\)](#).

**Administrador de IAM**: si es un administrador de IAM, es posible que quiera conocer información sobre cómo escribir políticas para administrar el acceso a Amazon Rekognition. Para consultar ejemplos de políticas basadas en la identidad de Amazon Rekognition que puede utilizar en IAM, consulte [Ejemplos de políticas de Amazon Rekognition basadas en identidades \(p. 493\)](#).

## Autenticación con identidades

La autenticación es la manera de iniciar sesión en AWS mediante credenciales de identidad. Para obtener más información acerca de cómo iniciar sesión con la AWS Management Console, consulte [Inicio de sesión en la AWS Management Console como usuario de IAM o usuario raíz](#) en la Guía del usuario de IAM.

Debe estar autenticado (haber iniciado sesión en AWS) como el usuario raíz de la Cuenta de AWS, como un usuario de IAM o asumiendo un rol de IAM. También puede utilizar la autenticación de inicio de sesión único de la empresa o incluso iniciar sesión con Google o Facebook. En estos casos, su administrador habrá configurado previamente la federación de identidad mediante roles de IAM. Cuando obtiene acceso a AWS mediante credenciales de otra empresa, asume un rol indirectamente.

Para iniciar sesión directamente en la [AWS Management Console](#), utilice la contraseña con su dirección de email de usuario raíz o con su nombre de usuario de IAM. Puede acceder a AWS mediante programación utilizando sus claves de acceso de usuario raíz o usuario de IAM. AWS proporciona SDK y herramientas de línea de comandos para firmar criptográficamente su solicitud con sus credenciales. Si no utiliza las herramientas de AWS, debe firmar usted mismo la solicitud. Para ello, utilice Signature Version 4, un protocolo para autenticar solicitudes de API de entrada. Para obtener más información acerca de cómo autenticar solicitudes, consulte [Proceso de firma de Signature Version 4](#) en la Referencia general de AWS.

Independientemente del método de autenticación que utilice, es posible que también deba proporcionar información de seguridad adicional. Por ejemplo, AWS le recomienda el uso de la autenticación multifactor (MFA) para aumentar la seguridad de su cuenta. Para obtener más información, consulte [Uso de la autenticación multifactor \(MFA\) en AWS](#) en la Guía del usuario de IAM.

## Cuenta de AWS usuario raíz

Cuando se crea por primera vez una Cuenta de AWS, se comienza con una única identidad de inicio de sesión que tiene acceso completo a todos los servicios y recursos de AWS de la cuenta. Esta identidad recibe el nombre de usuario raíz de la Cuenta de AWS y se accede a ella iniciando sesión con el email y la contraseña que utilizó para crear la cuenta. Recomendamos encarecidamente que no utilice el usuario raíz en sus tareas cotidianas, ni siquiera en las tareas administrativas. En lugar de ello, es mejor ceñirse a la [práctica recomendada de utilizar el usuario final exclusivamente para crear al primer usuario de IAM](#). A continuación, guarde las credenciales del usuario raíz en un lugar seguro y utilícelas tan solo para algunas tareas de administración de cuentas y servicios.

## Usuarios y grupos de IAM

Un [usuario de IAM](#) es una identidad de la Cuenta de AWS que dispone de permisos específicos para una sola persona o aplicación. Un usuario de IAM puede tener credenciales a largo plazo, como un nombre

de usuario y una contraseña o un conjunto de claves de acceso. Para obtener información sobre cómo generar claves de acceso, consulte [Administración de claves de acceso de los usuarios de IAM](#) en la Guía del usuario de IAM. Al generar claves de acceso para un usuario de IAM, asegúrese de ver y guardar de forma segura el par de claves. No puede recuperar la clave de acceso secreta en el futuro. En su lugar, debe generar un nuevo par de claves de acceso.

Un [grupo de IAM](#) es una identidad que especifica un conjunto de usuarios de IAM. No puede iniciar sesión como grupo. Puede usar los grupos para especificar permisos para varios usuarios a la vez. Los grupos facilitan la administración de los permisos de grandes conjuntos de usuarios. Por ejemplo, podría tener un grupo cuyo nombre fuese IAMAdmins y conceder permisos a dicho grupo para administrar los recursos de IAM.

Los usuarios son diferentes de los roles. Un usuario se asocia exclusivamente a una persona o aplicación, pero la intención es que cualquier usuario pueda asumir un rol que necesite. Los usuarios tienen credenciales permanentes a largo plazo y los roles proporcionan credenciales temporales. Para obtener más información, consulte [Cuándo crear un usuario de IAM \(en lugar de un rol\)](#) en la Guía del usuario de IAM.

## IAM roles

Un [rol de IAM](#) es una identidad de la Cuenta de AWS que dispone de permisos específicos. Es similar a un usuario de IAM, pero no está asociado a una determinada persona. Puede asumir temporalmente un rol de IAM en la AWS Management Console [cambiando de roles](#). Puede asumir un rol llamando a una operación de la AWS CLI o de la API de AWS, o utilizando una URL personalizada. Para obtener más información acerca de los métodos para el uso de roles, consulte [Uso de roles de IAM](#) en la Guía del usuario de IAM.

Los roles de IAM con credenciales temporales son útiles en las siguientes situaciones:

- Permisos de usuario de IAM temporales: un usuario de IAM puede asumir un rol de IAM para recibir temporalmente permisos distintos que le permitan realizar una tarea concreta.
- Acceso de usuarios federados: en lugar de crear un usuario de IAM, puede utilizar identidades existentes de AWS Directory Service, del directorio de usuarios de su empresa o de un proveedor de identidades web. A estas identidades se les llama usuarios federados. AWS asigna una función a un usuario federado cuando se solicita acceso a través de un [proveedor de identidad](#). Para obtener más información acerca de los usuarios federados, consulte [Usuarios federados y roles](#) en la Guía del usuario de IAM.
- Acceso entre cuentas: puede utilizar un rol de IAM para permitir que alguien (una entidad principal de confianza) de otra cuenta acceda a los recursos de la cuenta. Los roles son la forma principal de conceder acceso entre cuentas. Sin embargo, con algunos servicios de AWS, puede asociar una política directamente a un recurso (en lugar de utilizar un rol como proxy). Para obtener información acerca de la diferencia entre los roles y las políticas basadas en recursos para el acceso entre cuentas, consulte [Cómo los roles de IAM difieren de las políticas basadas en recursos](#) en la Guía del usuario de IAM.
- Acceso entre servicios: algunos servicios de AWS utilizan características de otros servicios de AWS. Por ejemplo, cuando realiza una llamada en un servicio, es común que ese servicio ejecute aplicaciones en Amazon EC2 o almacene objetos en Amazon S3. Es posible que un servicio haga esto usando los permisos de la entidad principal, usando un rol de servicio o usando un rol vinculado a servicios.
  - Permisos principales: cuando utiliza un usuario o un rol de IAM para llevar a cabo acciones en AWS, se lo considera una entidad principal. Las políticas conceden permisos a una entidad principal. Cuando utiliza algunos servicios, es posible que realice una acción que desencadene otra acción en un servicio diferente. En este caso, debe tener permisos para realizar ambas acciones. Para ver si una acción requiere acciones dependientes adicionales en una política, consulte [Claves de condición, recursos y acciones de Amazon Rekognition](#) en la Referencia de autorizaciones de servicio.
- Rol de servicio: un rol de servicio es un [rol de IAM](#) que adopta un servicio para realizar acciones en su nombre. Un administrador de IAM puede crear, modificar y eliminar un rol de servicio desde IAM. Para obtener más información, consulte [Creación de un rol para delegar permisos a un servicio de AWS](#) en la Guía del usuario de IAM.

- Rol vinculado a un servicio: un rol vinculado a un servicio es un tipo de función del servicio que está vinculado a un servicio de AWS. El servicio puede asumir el rol para realizar una acción en su 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.
- Aplicaciones que se ejecutan en Amazon EC2: puede utilizar un rol de IAM que le permita administrar credenciales temporales para las aplicaciones que se ejecutan en una instancia EC2 y realizan solicitudes a la AWS CLI o a la API de AWS. Es preferible hacerlo de este modo a almacenar claves de acceso en la instancia EC2. Para asignar un rol de AWS a una instancia EC2 y ponerla a disposición de todas las aplicaciones, cree un perfil de instancia asociado a la misma. Un perfil de instancia contiene el rol y permite a los programas que se ejecutan en la instancia EC2 obtener credenciales temporales. Para obtener más información, consulte [Uso de un rol de IAM para conceder permisos a aplicaciones que se ejecutan en instancias Amazon EC2](#) en la Guía del usuario de IAM.

Para obtener información sobre el uso de los roles de IAM, consulte [Cuándo crear un rol de IAM \(en lugar de un usuario\)](#) en la Guía del usuario de IAM.

## Administración de acceso mediante políticas

Para controlar el acceso en AWS, se crean políticas y se adjuntan a identidades de IAM o recursos de AWS. Una política es un objeto de AWS que, cuando se asocia a una identidad o un recurso, define sus permisos. Puede iniciar sesión como usuario raíz o usuario de IAM o puede asumir un rol de IAM. Cuando realiza una solicitud, AWS evalúa las políticas relacionadas basadas en identidades o en recursos. Los permisos en las políticas determinan si la solicitud se permite o se deniega. Las mayoría de las políticas se almacenan en AWS como documentos JSON. Para obtener más información sobre la estructura y el contenido de los documentos de política JSON, consulte [Información general de políticas JSON](#) en la Guía del usuario de IAM.

Los administradores pueden utilizar las políticas JSON de AWS para especificar quién tiene acceso a qué. Es decir, qué entidad principal puede realizar acciones en qué recursos y bajo qué condiciones.

Cada entidad de IAM (usuario o rol) comienza sin permisos. En otras palabras, de forma predeterminada, los usuarios no pueden hacer nada, ni siquiera cambiar sus propias contraseñas. Para conceder permiso a un usuario para hacer algo, el administrador debe asociarle una política de permisos. O bien el administrador puede agregar al usuario a un grupo que tenga los permisos necesarios. Cuando el administrador concede permisos a un grupo, todos los usuarios de ese grupo obtienen los permisos.

Las políticas de IAM definen permisos para una acción independientemente del método que se utilice para realizar la operación. Por ejemplo, suponga que dispone de una política que permite la acción `iam:GetRole`. Un usuario con dicha política puede obtener información del usuario de la AWS Management Console, la AWS CLI o la API de AWS.

## Uso de políticas basadas en identidades

Las políticas basadas en identidades son documentos de políticas de permisos JSON que puede asociar a una identidad, como un usuario de IAM, un grupo de usuarios o un rol. Estas políticas controlan qué acciones pueden realizar los usuarios y los roles, en qué recursos y bajo qué condiciones. Para obtener más información sobre cómo crear una política basada en identidades, consulte [Creación de políticas de IAM](#) en la Guía del usuario de IAM.

Las políticas basadas en identidad pueden clasificarse además como políticas insertadas o políticas administradas. Las políticas insertadas se integran directamente en un único usuario, grupo o rol. Las políticas administradas son políticas independientes que puede asociar a varios usuarios, grupos y roles de su Cuenta de AWS. Las políticas administradas incluyen las políticas administradas por AWS y las políticas administradas por el cliente. Para obtener más información acerca de cómo elegir una política administrada o una política insertada, consulte [Elegir 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. Ejemplos de políticas basadas en recursos son las políticas de confianza de roles de IAM y las políticas de bucket de Amazon S3. En los servicios que admiten políticas basadas en recursos, los administradores de servicios pueden utilizarlos para controlar el acceso a un recurso específico. Para el recurso al que se asocia la política, la política define qué acciones puede realizar una entidad principal especificada en ese recurso y en qué condiciones. Debe [especificar una entidad principal](#) en una política basada en recursos. Las entidades principales pueden incluir cuentas, usuarios, roles, usuarios federados o servicios de AWS.

Las políticas basadas en recursos son políticas insertadas que se encuentran en ese servicio. No se puede utilizar políticas de IAM administradas por AWS en una política basada en recursos.

## Listas de control de acceso (ACL)

Las listas de control de acceso (ACL) controlan qué entidades principales (miembros de cuentas, usuarios o roles) tienen permisos para acceder a un recurso. Las ACL son similares a las políticas basadas en recursos, aunque no utilizan el formato de documento de política JSON.

Amazon S3, AWS WAF y Amazon VPC son ejemplos de servicios que admiten las ACL. Para obtener más información sobre las ACL, consulte [Información general de Lista de control de acceso \(ACL\)](#) en la Guía para desarrolladores de Amazon Simple Storage Service.

## Otros tipos de políticas

AWS admite otros tipos de políticas adicionales menos frecuentes. Estos tipos de políticas pueden establecer el máximo de permisos que los tipos de políticas más frecuentes le otorgan.

- Límites de permisos: un límite de permisos es una característica avanzada que le permite establecer los permisos máximos que una política basada en identidades puede conceder a una entidad de IAM (usuario o rol de IAM). Puede establecer un límite de permisos para una identidad. Los permisos resultantes son la intersección de las políticas basadas en identidades de la entidad y los límites de sus permisos. Las políticas basadas en recursos que especifiquen el usuario o rol en el campo Principal no estarán restringidas por el límite de permisos. Una denegación explícita en cualquiera de estas políticas anulará el permiso. Para obtener más información sobre los límites de los permisos, consulte [Límites de permisos para las entidades de IAM](#) en la Guía del usuario de IAM.
- Políticas de control de servicio (SCP): las SCP son políticas de JSON que especifican los permisos máximos de una organización o una unidad organizativa (OU) en AWS Organizations. AWS Organizations es un servicio que le permite agrupar y administrar de manera centralizada varias Cuentas de AWS que posea su empresa. Si habilita todas las características en una organización, entonces podrá aplicar políticas de control de servicio (SCP) a una o todas sus cuentas. Las SCP limitan los permisos de las entidades de las cuentas miembro, incluido cada usuario raíz de la Cuenta de AWS. Para obtener más información acerca de Organizations y las SCP, consulte [Funcionamiento de las SCP](#) en la Guía del usuario de AWS Organizations.
- Políticas de sesión: las políticas de sesión son políticas avanzadas que se pasan como parámetro cuando se crea una sesión temporal mediante programación para un rol o un usuario federado. Los permisos de la sesión resultantes son la intersección de las políticas basadas en identidades del rol y las políticas de la sesión. Los permisos también pueden proceder de una política basada en recursos. Una denegación explícita en cualquiera de estas políticas anulará el permiso. 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 obtener información acerca de cómo AWS decide si permitir o no una solicitud cuando

hay varios tipos de políticas implicados, consulte [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 se encuentran disponibles con Amazon Rekognition. Para obtener una perspectiva general de cómo Amazon Rekognition y otros servicios funcionan con IAM, consulte [AWS Services that Function with IAM](#) en la IAM User Guide.

### Temas

- [Políticas de Amazon Rekognition basadas en identidades \(p. 486\)](#)
- [Políticas de Amazon Rekognition basadas en recursos \(p. 487\)](#)
- [Roles de IAM de Amazon Rekognition \(p. 488\)](#)

## Políticas de Amazon Rekognition basadas en identidades

Con las políticas basadas en identidades de IAM, puede especificar las acciones y 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 utilizar las políticas JSON de AWS para especificar quién tiene acceso a qué. Es decir, qué entidad principal puede llevar a cabo acciones en qué recursos y bajo qué condiciones.

El elemento `Action` de una política JSON describe las acciones que puede utilizar para permitir o denegar el acceso en una política. Las acciones de la política generalmente tienen el mismo nombre que la operación de API de AWS asociada. Hay algunas excepciones, como acciones de solo permiso que no tienen una operación de API coincidente. También hay algunas operaciones que requieren varias acciones en una política. Estas acciones adicionales se denominan acciones dependientes.

Incluya acciones en una política para conceder permisos y así llevar a cabo la operación asociada.

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 Amazon RekognitionDeleteLabelsOperación de la API, incluye `rekognition:DetectLabels` acción 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 comienzan 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 IAM User Guide.

## Recursos

Los administradores pueden utilizar las políticas JSON de AWS para especificar quién tiene acceso a qué. Es decir, qué entidad principal puede realizar acciones en qué recursos y bajo qué condiciones.

El elemento **Resource** de la política JSON especifica el objeto u objetos a los que se aplica la acción. Las instrucciones deben contener un elemento **Resource** o **NotResource**. Como práctica recomendada, especifique un recurso utilizando el [Nombre de recurso de Amazon \(ARN\)](#). Puede hacerlo para acciones que admitan un tipo de recurso específico, conocido como permisos de nivel de recurso.

Para las acciones que no admiten permisos de nivel de recurso, como las operaciones de descripción, utilice un carácter comodín (\*) para indicar que la instrucción se aplica a todos los recursos.

```
"Resource": "*"
```

Para obtener más información acerca del formato de los ARN, consulte [Nombres de recursos de Amazon \(ARN\) y espacios de nombres de servicios de AWS](#).

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 y sus ARN, consulte [Recursos definidos por Amazon Rekognition](#) en la [Guía del usuario de IAM](#). Para obtener información acerca de con qué acciones puede especificar los 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 globales de AWS, consulte [Claves de contexto de condición globales de AWS](#) en la [Guía del usuario de IAM](#).

## Políticas de Amazon Rekognition basadas en recursos

Amazon Rekognition no admite políticas basadas en recursos.

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 `Tests3bucket`.

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Allow",
 "Action": "s3:GetObject",
 "Resource": [
 "arn:aws:s3:::Tests3bucket/*"
]
 }
]
}
```

Para usar un bucket de S3 con el control de versiones habilitado, añada la acción `s3:GetObjectVersion`, como se muestra en el siguiente ejemplo.

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Allow",
 "Action": [
 "s3:GetObject",
 "s3:GetObjectVersion"
],
 "Resource": [
 "arn:aws:s3:::Tests3bucket/*"
]
 }
]
}
```

## Roles de IAM de Amazon Rekognition

Un [rol de IAM](#) es una entidad de la cuenta de AWS que dispone de 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. Las credenciales de seguridad temporales se obtienen mediante una llamada a operaciones de la API de AWS STS, como [AssumeRole](#) o [GetFederationToken](#).

Amazon Rekognition admite el uso de credenciales temporales.

### Roles vinculados a servicios

Los [roles vinculados a servicios](#) permiten a los servicios de AWS obtener acceso a los recursos de otros servicios para completar una acción en su 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.

## Elegir un rol de IAM en Amazon Rekognition

Al configurar Amazon Rekognition para analizar vídeos almacenados, debe elegir un rol para permitir 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” (p. 70).

## AWS Políticas administradas por para Amazon Rekognition

Para agregar permisos a usuarios, grupos y roles, es más fácil utilizar las políticas administradas por AWS que escribirlas uno mismo. Se necesita tiempo y experiencia para [crear políticas de IAM administradas por el cliente](#) que proporcionen a su equipo solo los permisos necesarios. Para comenzar a hacerlo con rapidez, puede utilizar nuestras políticas administradas por AWS. Estas políticas cubren casos de uso comunes y están disponibles en su cuenta de AWS. Para obtener más información sobre las políticas administradas por AWS, consulte [Políticas administradas por AWS](#) en la Guía del usuario de IAM.

Los servicios de AWS mantienen y actualizan las políticas administradas por AWS. No puede cambiar los permisos en las políticas administradas por AWS. En ocasiones, los servicios agregan permisos adicionales a una política administrada por 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 por AWS cuando se lanza una nueva característica o cuando se ponen a disposición nuevas operaciones. Los servicios no quitan permisos de una política administrada por AWS, por lo que las actualizaciones de políticas no deteriorarán los permisos existentes.

Además, AWS admite políticas administradas para funciones de trabajo que abarcan varios servicios. Por ejemplo, la política administrada por ReadOnlyAccessAWS proporciona acceso de solo lectura a todos los servicios y los recursos de AWS. Cuando un servicio lanza una nueva característica, AWS agrega permisos de solo lectura para las operaciones y los recursos nuevos. Para obtener una lista y descripciones de las políticas de funciones de trabajo, consulte [Políticas administradas por AWS para funciones de trabajo](#) en la Guía del usuario de IAM.

## Política administrada de AWS: AmazonRekognitionFullAccess

AmazonRekognitionFullAccess concede acceso completo a los recursos de Amazon Rekognition, incluida la creación y eliminación de colecciones.

Puede adjuntar la política AmazonRekognitionFullAccess a las identidades de IAM.

### Detalles sobre los permisos

Esta política incluye los siguientes permisos.

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

```
]
}
```

## Política administrada de AWS: AmazonRekognitionReadOnlyAccess

AmazonRekognitionReadOnlyAccess concede acceso de solo lectura a los recursos de Amazon Rekognition.

Puede adjuntar la política AmazonRekognitionReadOnlyAccess a las identidades de IAM.

Detalles sobre los permisos

Esta política incluye los siguientes permisos.

```
{
 "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:DetectCustomLabel",
 "rekognition:DetectProtectiveEquipment",
 "rekognition>ListTagsForResource",
 "rekognition>ListDatasetEntries",
 "rekognition>ListDatasetLabels",
 "rekognition:DescribeDataset"
],
 "Resource": "*"
 }
]
}
```

## Política administrada de AWS: AmazonRekognitionServiceRole

AmazonRekognitionServiceRole permite a Amazon Rekognition llamar a Amazon Kinesis Data Streams y a los servicios de Amazon SNS en su nombre.

Puede adjuntar la política AmazonRekognitionServiceRole a las identidades de IAM.

#### Detalles sobre los permisos

Esta política incluye los siguientes permisos.

```
{
 "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 administrada de AWS: [AmazonRekognitionCustomLabelsFullAccess](#)

Esta política es para los usuarios de Etiquetas personalizadas de Amazon Rekognition. Utilice la política AmazonRekognitionCustomLabelsFullAccess para permitir a los usuarios acceso completo a la API de etiquetas personalizadas de Amazon Rekognition y acceso completo a los depósitos de consola creados por la consola de etiquetas personalizadas de Amazon Rekognition.

#### Detalles sobre los permisos

Esta política incluye los siguientes permisos.

```
{
 "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:CreateProject",
 "rekognition:CreateProjectVersion",
 "rekognition:StartProjectVersion",
 "rekognition:StopProjectVersion",
 "rekognition:DescribeProjects",
 "rekognition:DescribeProjectVersions",
 "rekognition:DetectCustomLabelLabels",
 "rekognition:DeleteProject",
 "rekognition:DeleteProjectVersion"
 "rekognition:TagResource",
 "rekognition:UntagResource",
 "rekognition>ListTagsForResource",
 "rekognition:CreateDataset",
 "rekognition>ListDatasetEntries",
 "rekognition>ListDatasetLabels",
 "rekognition:DescribeDataset",
 "rekognition:UpdateDatasetEntries",
 "rekognition:DistributeDatasetEntries",
 "rekognition:DeleteDataset"
],
 "Resource": "*"
 }
]
}

```

## Actualizaciones de Amazon Rekognition aAWSpolíticas administradas

Muestra los detalles de las actualizaciones deAWSpolíticas administradas por para Amazon Rekognition debido a que este servicio comenzó a realizar el seguimiento de estos cambios. Para obtener alertas automáticas sobre cambios en esta página, suscríbase a la fuente RSS en Amazon RekognitionHistorial de revisión(Se ha creado el certificado).

Cambio	Descripción	Fecha
Actualización de la administración de conjuntos de datos para las siguientes políticas administradas:	<ul style="list-style-type: none"> <li>• <a href="#">Política administrada de AWS: AmazonRekognitionReadOnlyAccess (p. 490)</a></li> <li>• <a href="#">Política administrada de AWS: AmazonRekognitionFullAccess (p. 489)</a></li> <li>• <a href="#">Política administrada de AWS: AmazonRekognitionCustomLabelLabelsFullAccess (p. 491)</a></li> </ul>	1 de noviembre de 2021

Cambio	Descripción	Fecha
Actualización de etiquetado para <a href="#">Política administrada de AWS: AmazonRekognitionReadOnlyAccess</a> ( <a href="#">AmazonRekognitionFullAccess</a> y <a href="#">AmazonRekognitionReadOnlyAccess</a> )	Amazon Rekognition ha añadido nuevas acciones de etiquetado a las políticas administradas de AWS: <a href="#">AmazonRekognitionFullAccess</a> y <a href="#">AmazonRekognitionReadOnlyAccess</a> .	2 de abril de 2021
Amazon Rekognition comenzó a realizar seguimientos de los cambios	Amazon Rekognition comenzó a realizar seguimientos de los cambios para su AWSpolíticas administradas.	2 de abril de 2021

## Ejemplos de políticas de Amazon Rekognition basadas en identidades

De forma predeterminada, los usuarios y roles de IAM no tienen permiso para crear ni modificar recursos de Amazon Rekognition. Tampoco pueden realizar tareas mediante la AWS Management Console, la AWS CLI, o la API de AWS. 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 adjuntar esas políticas a los usuarios o grupos de IAM que necesiten esos permisos.

Para obtener información acerca de cómo crear una política basada en identidades 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

- [Prácticas recomendadas relativas a políticas \(p. 493\)](#)
- [Uso de la consola de Amazon Rekognition \(p. 494\)](#)
- [Ejemplo de políticas de etiquetas personalizadas de Amazon Rekognition \(p. 494\)](#)
- [Ejemplo 1: Permitir a un usuario acceso de solo lectura a los recursos de \(p. 494\)](#)
- [Ejemplo 2: Permitir a un usuario acceso completo a los recursos de \(p. 495\)](#)
- [Permitir a los usuarios consultar sus propios permisos \(p. 495\)](#)

## Prácticas recomendadas relativas a políticas

Las políticas basadas en identidades son muy eficaces. Determinan si alguien puede crear, acceder o eliminar los recursos de Amazon Rekognition de su cuenta. Estas acciones pueden generar costes adicionales para su Cuenta de AWS. Siga estas directrices y recomendaciones al crear o editar políticas basadas en identidades:

- Aprenda a usarAWSpolíticas administradas— Para empezar a utilizar Amazon Rekognition rápidamente, utiliceAWSpolíticas administradas por para proporcionar a los empleados los permisos necesarios. Estas políticas ya están disponibles en su cuenta, y las mantiene y actualiza AWS. Para obtener más información, consulte [Introducción sobre el uso de permisos con políticas administradas por AWS](#) en la Guía del usuario de IAM.
- Conceder privilegios mínimos: al crear políticas personalizadas, conceda solo los permisos necesarios para llevar a cabo una tarea. Comience con un conjunto mínimo de permisos y conceda permisos adicionales según sea necesario. Por lo general, es más seguro que comenzar con permisos que son

demasiado tolerantes e intentar hacerlos más estrictos más adelante. Para obtener más información, consulte [Conceder privilegios mínimos](#) en la Guía del usuario de IAM.

- Habilitar la MFA para operaciones confidenciales: para mayor seguridad, obligue a los usuarios de IAM a utilizar la autenticación multifactor (MFA) para acceder a recursos u operaciones de API confidenciales. Para obtener más información, consulte [Uso de la autenticación multifactor \(MFA\) en AWS](#) en la Guía del usuario de IAM.
  - Utilizar condiciones de política para mayor seguridad: en la medida en que sea práctico, defina las condiciones en las que las políticas basadas en identidades permitan el acceso a un recurso. Por ejemplo, puede escribir condiciones para especificar un rango de direcciones IP permitidas desde el que debe proceder una solicitud. También puede escribir condiciones para permitir solicitudes solo en un intervalo de hora o fecha especificado o para solicitar el uso de SSL o MFA. Para obtener más información, consulte [Elementos de la política de JSON de IAM: Condición](#)en la IAM User Guide.

## 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 utilizar la consola de Amazon Rekognition. Para obtener información sobre las etiquetas personalizadas de Amazon Rekognition, consulte [Paso 5: Configuración de permisos de la consola de etiquetas personalizadas de Amazon Rekognition](#).

No es necesario que conceda permisos mínimos para la consola a los usuarios que solo realizan llamadas a la AWS CLI o a la API de AWS. En su lugar, permite acceso únicamente a las acciones que coincidan con la operación de API que intenta realizar.

## 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 a un usuario acceso de solo lectura a los recursos de

En el siguiente ejemplo se concede acceso de solo lectura a los recursos de Amazon Rekognition.

```
 "rekognition:GetFaceSearch",
 "rekognition:GetTextDetection",
 "rekognition:GetSegmentDetection",
 "rekognition:DescribeStreamProcessor",
 "rekognition>ListStreamProcessors",
 "rekognition:DescribeProjects",
 "rekognition:DescribeProjectVersions",
 "rekognition:DetectCustomLabelLabels",
 "rekognition:DetectProtectiveEquipment",
 "rekognition>ListTagsForResource",
 "rekognition>ListDatasetEntries",
 "rekognition>ListDatasetLabels",
 "rekognition:DescribeDataset"

],
 "Resource": "*"
}
]
}
```

## Ejemplo 2: Permitir a un usuario acceso completo a los recursos de

En el siguiente ejemplo se concede acceso completo a los recursos de Amazon Rekognition.

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

## 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 llevar a cabo esta acción en la consola o mediante programación con la AWS CLI o la API de 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": "*"
}
]
}
```

## 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 \(p. 496\)](#)
- [No tengo autorización para realizar la operación iam:PassRole \(p. 496\)](#)
- [Quiero ver mis claves de acceso \(p. 497\)](#)
- [Soy administrador y deseo permitir que otros accedan a Amazon Rekognition \(p. 497\)](#)
- [Quiero permitir a la gente fuera de miAWScuenta para acceder a mis recursos de Amazon Rekognition \(p. 497\)](#)

## No tengo autorización para realizar una acción en Amazon Rekognition

Si la AWS Management Console le indica que no está autorizado para llevar a cabo una acción, debe ponerse en contacto con su administrador para recibir ayuda. Su administrador es la persona que le facilitó su nombre de usuario y contraseña.

En el siguiente ejemplo, el error se produce cuando el usuario `mateojackson` de IAM, intenta utilizar la consola para ver detalles sobre un [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 tengo autorización para realizar la operación iam:PassRole

Si recibe un error que indica que no está autorizado para llevar a cabo la acción `iam:PassRole`, debe ponerse en contacto con su administrador para recibir ayuda. Su administrador es la persona que le facilitó su nombre de usuario y contraseña. Pida a la persona que actualice las políticas de forma que pueda transferir un rol a Amazon Rekognition.

Algunos servicios de AWS le permiten transferir un rol existente a dicho servicio en lugar de crear un nuevo rol de servicio o uno vinculado al servicio. Para ello, debe tener permisos para transferir el rol al servicio.

En el siguiente ejemplo, el error se produce cuando un usuario de IAM denominado **MaryMajor** intenta utilizar la consola para realizar una acción en Amazon Rekognition. Sin embargo, la acción requiere que el servicio cuente con permisos otorgados por 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, Mary pide a su administrador que actualice sus políticas para que pueda realizar la acción `iam:PassRole`.

## Quiero ver mis claves de acceso

Después de crear sus claves de acceso de usuario de IAM, puede ver su ID de clave de acceso en cualquier momento. Sin embargo, no puede volver a ver su clave de acceso secreta. Si pierde la clave de acceso secreta, debe crear un nuevo par de claves de acceso.

Las claves de acceso se componen de dos partes: un ID de clave de acceso (por ejemplo, `AKIAIOSFODNN7EXAMPLE`) y una clave de acceso secreta (por ejemplo, `wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY`). El ID de clave de acceso y la clave de acceso secreta se utilizan juntos, como un nombre de usuario y contraseña, para autenticar sus solicitudes. Administre sus claves de acceso con el mismo nivel de seguridad que para el nombre de usuario y la contraseña.

### Important

No proporcione las claves de acceso a terceros, ni siquiera para que le ayuden a [buscar el ID de usuario canónico](#). Si lo hace, podría conceder a otra persona acceso permanente a su cuenta.

Cuando cree un par de claves de acceso, se le pide que guarde el ID de clave de acceso y la clave de acceso secreta en un lugar seguro. La clave de acceso secreta solo está disponible en el momento de su creación. Si pierde la clave de acceso secreta, debe agregar nuevas claves de acceso a su usuario de IAM. Puede tener un máximo de dos claves de acceso. Si ya cuenta con dos, debe eliminar un par de claves antes de crear uno nuevo. Para consultar las instrucciones, consulte [Administración de claves de acceso](#) en la Guía del usuario de IAM.

## Soy administrador y deseo permitir que otros accedan a Amazon Rekognition

Para permitir que otros accedan a Amazon Rekognition, debe crear una entidad de IAM (usuario o rol) para la persona o aplicación que necesita acceso. Esta persona utilizará las credenciales de la entidad para acceder a AWS. A continuación, debe adjuntar una política a la entidad que les conceda los permisos correctos en Amazon Rekognition.

Para comenzar de inmediato, consulte [Creación del primer grupo y usuario delegado de IAM](#) en la Guía del usuario de IAM.

## Quiero permitir a la gente fuera de mi AWS cuenta para acceder a mis recursos de Amazon Rekognition

Puede crear un rol que los usuarios de otras cuentas o las personas externas a la organización puedan utilizar para acceder a sus recursos. Puede especificar una persona de confianza para que asuma el rol. En el caso de los servicios que admitan las políticas basadas en recursos o las listas de control de acceso (ACL), puede utilizar dichas políticas para conceder a las personas acceso a sus recursos.

Para obtener más información, consulte lo siguiente:

- Para obtener información acerca de si Amazon Rekognition admite estas características, consulte [Cómo funciona Amazon Rekognition con IAM \(p. 486\)](#).
- Para obtener información acerca de cómo proporcionar acceso a los recursos de las Cuentas de AWS de su propiedad, consulte [Proporcionar acceso a un usuario de IAM a otra Cuenta de AWS de la que es propietario](#) en la Guía del usuario de IAM.
- Para obtener información acerca de cómo proporcionar acceso a los recursos a Cuentas de AWS de terceros, consulte [Proporcionar acceso a Cuentas de AWS que son propiedad de terceros](#) en la Guía del usuario de IAM.
- Para obtener información sobre cómo proporcionar acceso mediante una identidad federada, consulte [Proporcionar acceso a usuarios autenticados externamente \(identidad federada\)](#) en la Guía del usuario de IAM.
- Para obtener información sobre la diferencia entre los roles y las políticas basadas en recursos para el acceso entre cuentas, consulte [Cómo los roles de IAM difieren de las políticas basadas en recursos](#) en la Guía del usuario de IAM.

## Protección de datos en Amazon Rekognition

La [AWS Modelo de responsabilidad compartida](#) 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. Usted es responsable de mantener el control sobre el contenido alojado en esta infraestructura. Este contenido incluye la configuración de seguridad y las tareas de administración de los servicios de AWS que usted utiliza. Para obtener más información sobre la privacidad de los datos, consulte las [Preguntas frecuentes sobre la privacidad de datos](#). Para obtener información sobre la protección de datos en Europa, consulte la publicación de blog [AWS Shared Responsibility Model and GDPR](#) en el Blog de seguridad de AWS.

Para fines de protección de datos, recomendamos proteger las credenciales de Cuenta de AWS y configurar cuentas de usuario individuales con AWS Identity and Access Management (IAM). De esta manera, solo se otorgan a cada usuario los permisos necesarios para cumplir con sus obligaciones laborales. También recomendamos proteger sus datos de las siguientes formas:

- Utilice Multi-Factor Authentication (MFA) con cada cuenta.
- Utilice SSL/TLS para comunicarse con los recursos de AWS. Recomendamos TLS 1.2 o una versión posterior.
- Configure la API y el registro de actividad del usuario con AWS CloudTrail.
- Utilice las soluciones de cifrado de AWS, junto con todos los controles de seguridad predeterminados dentro de los servicios de AWS.
- Utilice avanzados servicios de seguridad administrados, como Amazon Macie, que lo ayuden a detectar y proteger los datos personales almacenados en Amazon S3.
- Si necesita módulos criptográficos validados FIPS 140-2 al acceder a AWS a través de una interfaz de línea de comandos o una API, utilice un punto de enlace de FIPS. Para obtener más información sobre los puntos de enlace de FIPS disponibles, consulte [Estándar de procesamiento de la información federal \(FIPS\) 140-2](#).

Recomendamos encarecidamente que nunca introduzca información de identificación confidencial, como, por ejemplo, direcciones de email de sus clientes, en etiquetas o en los campos de formato libre, como el campo Name (Nombre). Esto incluye cuando trabaje con Rekognition u otro AWS servicio que utilizan la consola, la API, AWS CLI, o bien AWSSDK. Los datos que ingresa en etiquetas o campos de formato libre utilizados para los nombres se pueden utilizar 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 sus datos.

### Cifrado en reposo

#### Amazon Rekognition Image

##### Imágenes

Las imágenes pasadas a operaciones de la API de Amazon Rekognition pueden almacenarse y utilizarse para mejorar el servicio a no ser que haya optado por no ser que haya optado por no darse de [Página de políticas de exclusión de servicios de IA](#) siguiendo el proceso explicado 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. Los datos no tienen sentido por sí solos, actúan eficazmente como un hash y no se les puede aplicar ingeniería inversa. Los datos no pasan por ningún otro proceso de cifrado adicional.

#### Amazon Rekognition Video

##### Videos

Para analizar un vídeo, Amazon Rekognition copia sus vídeos en el servicio para su procesamiento. El vídeo puede almacenarse y utilizarse para mejorar el servicio a no ser que haya optado por no ser que haya optado por no visitar el [Página de políticas de exclusión de servicios de IA](#) siguiendo el proceso explicado allí. Los vídeos se cifran en reposo (Amazon S3) mediante AWS Key Management Service (SSE-KMS).

#### Etiquetas personalizadas de Amazon Rekognition

Las etiquetas personalizadas de Amazon Rekognition Cifran sus datos en reposo.

##### Imágenes

Para entrenar a su modelo, Amazon Rekognition Custom Labels hace una copia de las imágenes de prueba y formación de origen. Las imágenes copiadas se cifran en reposo en Amazon Simple Storage Service (S3) mediante cifrado del lado del servidor con unAWS KMS key que proporciona o unAWSClave de KMS propiedad. Las etiquetas personalizadas de Amazon Rekognition solo admiten claves de KMS simétricas. Las imágenes de origen no se ven afectadas. Para obtener más información, consulte [Formación de un modelo de etiquetas personalizadas de Amazon Rekognition](#).

##### Modelos

De forma predeterminada, las etiquetas personalizadas de Amazon Rekognition Cifran los modelos entrenados y los archivos de manifiesto almacenados en los buckets de Amazon S3 mediante cifrado del lado del servidor mediante un cifrado del lado del servidor mediante unClave propiedad de AWS. Para obtener más información, consulte [Protección de los datos con el cifrado del lado del servidor](#). Los resultados de formación se escriben en el bucket de especificado en el `OutputConfig` parámetro de entrada a [the section called “CreateProjectVersion” \(p. 532\)](#). Los resultados de la formación se cifran mediante la configuración de cifrado configurada para el bucket (`OutputConfig`).

## Cucharón consola

La consola de etiquetas personalizadas de Amazon Rekognition crea un depósito de Amazon S3 (depósito de consola) que puede utilizar para administrar sus proyectos. El depósito de la consola se cifra mediante el cifrado predeterminado de Amazon S3. Para obtener más información, consulte [Cifrado predeterminado de Amazon Simple Storage Service para los buckets de S3](#). Si utiliza su propia clave KMS, configure el depósito de la consola después de crearlo. Para obtener más información, consulte [Protección de los datos con el cifrado del lado del servidor](#). Las etiquetas personalizadas de Amazon Rekognition bloquean el acceso público al depósito de la consola.

## Cifrado en tránsito

Los puntos de enlace 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 [Conceptos de AWS Key Management Service](#).

## Privacidad del tráfico entre redes

Un punto de enlace 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 enlace de la VPC](#) en la Guía del usuario de Amazon VPC. Para obtener información sobre el uso de endpoints de Amazon VPC con Amazon Rekognition, consulte [Uso de Amazon Rekognition con endpoints de Amazon VPC \(p. 507\)](#).

## Monitorización Rekognition

Con CloudWatch, puede obtener métricas de las distintas operaciones de Rekognition o métricas globales de Rekognition para su cuenta. Puede usar las métricas para realizar un seguimiento del estado de la solución basada en Rekognition y configurar alarmas para que se le notifique cuando una o varias métricas queden fuera del 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, puede usar [Amazon CloudWatch](#), [AmazonAWS Command Line Interface](#), o el [API de CloudWatch](#).

También puede ver métricas globales durante un periodo de tiempo seleccionado mediante la consola de Rekognition. Para obtener más información, consulte [Ejercicio 4: Ver métricas globales \(consola\) \(p. 26\)](#).

## Uso de métricas de CloudWatch 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 `Operación`. 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 de Rekognition mediante la AWS Management Console, el AWS CLI o API de CloudWatch. También puede utilizar la API de CloudWatch API mediante uno de los kits de desarrollo de software (SDK) de Amazon AWS o las herramientas de la API de CloudWatch. La consola muestra una serie de gráficos basados en los datos sin procesar de la API de CloudWatch. 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 <code>Sum</code> de la métrica <code>DetectedFaceCount</code> .
¿Cómo puedo saber si mi aplicación ha alcanzado el número máximo de solicitudes por segundo?	Monitorice la estadística <code>Sum</code> de la métrica <code>ThrottledCount</code> .
¿Cómo puedo monitorizar los errores de solicitud?	Utilice la estadística <code>Sum</code> de la métrica <code>UserErrorCount</code> .
¿Cómo puedo encontrar el número total de solicitudes?	Utilice las estadísticas <code>ResponseTime</code> y <code>Data Samples</code> de la métrica <code>ResponseTime</code> . 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 <code>SuccessfulRequestCount</code> .
¿Cómo puedo monitorizar la latencia de las llamadas a operaciones de Rekognition?	Utilice la métrica <code>ResponseTime</code> .
¿Cómo puedo monitorizar cuántas veces <code>IndexFaces</code> se han añadido caras correctamente a las colecciones de Rekognition?	Monitorice la estadística <code>Sum</code> con la métrica <code>SuccessfulRequestCount</code> y la operación <code>IndexFaces</code> . Utilice la dimensión <code>Operation</code> para seleccionar la operación y la métrica.

Debe disponer de los permisos de CloudWatch adecuados para monitorear Rekognition con CloudWatch. Para obtener más información, consulte [Autenticación y control de acceso de Amazon CloudWatch](#).

## Acceso a las métricas de reconocimiento

En los siguientes ejemplos se muestra cómo obtener acceso a las métricas de Rekognition mediante la consola de CloudWatch, el AWS CLI o API de CloudWatch.

### Para ver las métricas (consola)

1. Abra la consola de CloudWatch en <https://console.aws.amazon.com/cloudwatch/>.
2. Elija Metrics, elija la pestaña All Metrics y, a continuación, elija Rekognition.
3. Elija Metrics with no dimensions 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 la AWS CLI y escriba 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 tener acceso a las métricas (API de CloudWatch)

- Llame a [GetMetricStatistics](#). Para obtener más información, consulte la[Referencia del API de Amazon CloudWatch](#).

## Crear una alarma

Puede crear una alarma de CloudWatch que envíe un mensaje de Amazon Simple Notification Service (Amazon SNS) cuando la alarma cambia de estado. Una alarma vigila una única métrica durante el periodo especificado y realiza una o varias acciones en función del valor de la métrica relativo a un determinado umbral durante una serie de periodos 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 para los cambios de estado prolongados. Las alarmas de CloudWatch no invocan acciones simplemente porque se encuentren 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 la AWS Management Console y abra la consola de CloudWatch en <https://console.aws.amazon.com/cloudwatch/>.
2. Elija Create Alarm. Esto lanza el Create Alarm Wizard (Asistente de creación de alarmas).
3. En la lista de métricas Metrics with no dimensions, elija Rekognition Metrics 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 Time Range, seleccione un valor de intervalo de fechas que incluya las operaciones de detección de rostros a las que ha llamado. Elija Next (Siguiente)
5. Rellene Name y Description. Para Whenever, elija  $\geq$  e introduzca un valor máximo de su elección.
6. Si desea que CloudWatch le envíe un correo electrónico cuando se alcance el estado de la alarma, paraSiempre que esta alarma:, eligeEl estado es ALARM. Para enviar alarmas a un tema de Amazon SNS existente, paraEnviar notificación a:, elija un tema de SNS existente. Para definir el nombre y las direcciones de correo electrónico para una nueva lista de suscripción de correo electrónico, elijaCrear tema deCloudWatch guarda la lista y la muestra en el campo para que pueda utilizarla para definir nuevas alarmas.

Note

Si usaCrear tema dePara 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 Alarm Preview. Elija Create Alarm.

#### Para configurar una alarma (AWS CLI)

- Abra la AWS CLI y escriba el siguiente comando. Cambiar el valor del `alarm-actions` para hacer referencia al tema de Amazon SNS que ha 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 (API de CloudWatch)

- Llame a [PutMetricAlarm](#). Para obtener más información, consulte [Referencia del API de Amazon CloudWatch](#).

## Métricas de CloudWatch para Rekognition

Esta sección contiene información acerca de las métricas de Amazon CloudWatch y el `OperaciónDimension` 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: Ver métricas globales \(consola\) \(p. 26\)](#).

## Métricas de CloudWatch para Rekognition

En la siguiente tabla se resumen las métricas de Rekognition.

Métrica	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: <code>Sum</code> , <code>Average</code>
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 <a href="#">Límites de los servicios de AWS</a> . Unidad: Recuento Estadísticas válidas: <code>Sum</code> , <code>Average</code>

Métrica	Descripción
ResponseTime	<p>El tiempo en milisegundos que tarda en que Rekognition calcula la respuesta.</p> <p>Unidades:</p> <ol style="list-style-type: none"><li>1. Recuento para la estadística Data Samples</li><li>2. Milisegundos para la estadística Average</li></ol> <p>Estadísticas válidas: Data Samples, Average</p> <p>Note</p> <p>La ResponseTimeLa métrica no está incluida en el panel de métricas de Rekognition.</p>
DetectedFaceCount	<p>El número de rostros detectados con la operación IndexFaces o DetectFaces.</p> <p>Unidad: Recuento</p> <p>Estadísticas válidas: Sum, Average</p>
DetectedLabelCount	<p>El número de etiquetas detectadas con la operación DetectLabels.</p> <p>Unidad: Recuento</p> <p>Estadísticas válidas: Sum, Average</p>
ServerErrorCount	<p>El número de errores de servidor. El intervalo de códigos de respuesta de un error de servidor comprende de 500 a 599.</p> <p>Unidad: Recuento</p> <p>Estadísticas válidas: Sum, Average</p>
UserErrorCount	<p>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.</p> <p>Unidad: Recuento</p> <p>Estadísticas válidas: Sum, Average</p>

## Dimensión de CloudWatch 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 acerca de las dimensiones, consulte [Dimensiones](#)en laGuía del usuario de Amazon CloudWatch.

# Registro de llamadas a la API de Amazon Rekognition conAWS CloudTrail

Amazon Rekognition se integra conAWS CloudTrail, un servicio que proporciona un registro de las acciones que realiza un usuario, un rol o unAWSservicio de Amazon Rekognition. CloudTrail captura todas

las llamadas a la API para 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 un registro de seguimiento, puede habilitar la entrega continua de eventos de CloudTrail a un bucket de Amazon S3, incluidos los eventos de Amazon Rekognition. Si no configura un registro de seguimiento, puede ver los eventos más recientes de la consola de CloudTrail en el Event history (Historial de eventos). Mediante la información recopilada por CloudTrail, puede determinar la solicitud que se realizó a Amazon Rekognition, la dirección IP de origen desde la que se realizó, quién la realizó, cuándo se realizó y otros detalles adicionales.

Para obtener más información acerca de CloudTrail, consulte la [AWS CloudTrailGuía del usuario de](#).

## Información de Amazon Rekognition en CloudTrail

CloudTrail se habilita en su cuenta de AWS cuando la crea. Cuando se produce una actividad en Amazon Rekognition, dicha actividad se registra en un evento de CloudTrail junto con los demásAWSEventos de servicio deHistorial de eventos. Puede ver, buscar y descargar los últimos eventos de la cuenta de AWS. Para obtener más información, consulte [Ver eventos con el historial de eventos de CloudTrail](#).

Para mantener un registro continuo de los eventos deAWSuenta, incluidos los eventos de Amazon Rekognition, cree un registro de seguimiento. Un registro de seguimiento permite a CloudTrail enviar 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. El registro de seguimiento registra los eventos de todas las regiones de la partición de AWS y envía los archivos de registro al bucket de Amazon S3 especificado. También es posible configurar otros servicios de AWS para analizar en profundidad y actuar en función de los datos de eventos recopilados en los registros de CloudTrail. Para obtener más información, consulte los siguientes temas:

- [Introducción a la creación de registros de seguimiento](#)
- [Servicios e integraciones compatibles con CloudTrail](#)
- [Configuración de notificaciones de Amazon SNS para CloudTrail](#)
- [Recibir archivos de registro de CloudTrail de varias regiones](#) y [Recibir archivos de registro de CloudTrail de varias cuentas](#)

CloudTrail registra todas las acciones de Amazon Rekognition y se documentan en[Referencia de la API de Amazon Rekognition](#). Por ejemplo, las llamadas a las acciones `CreateCollection`, `CreateStreamProcessor` y `DetectCustomLabel`s generan entradas en los archivos de registros de 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 AWS Identity and Access Management (IAM) o credenciales de usuario raíz.
- Si la solicitud se realizó con credenciales de seguridad temporales de un rol o fue un usuario federado.
- Si la solicitud la realizó otro servicio de AWS.

Para obtener más información, consulte el [Elemento userIdentity de CloudTrail](#).

## Descripción de las entradas de archivos de registro de Amazon Rekognition

Un registro de seguimiento es una configuración que permite la entrega de eventos como archivos de registros en un bucket de Amazon S3 que especifique. Los archivos log de CloudTrail pueden contener una o varias entradas de log. Un evento representa una solicitud específica realizada desde un origen

y contiene información sobre la acción solicitada, la fecha y la hora de la acción, los parámetros de la solicitud, etc. Los archivos de registro de CloudTrail no rastrean el orden en la pila de las llamadas públicas a la API, por lo que estas no aparecen en ningún orden específico.

En el siguiente ejemplo, se muestra una entrada de registro de CloudTrail con las acciones de la siguiente API:StartLabelDetectionyDetectLabels.

```
{
 "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": "653de5a7ee03bd5083edde98ea8fce5794fceaa66d077bdd4cfb39d71aff8fc25"
 },
 "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": "2020-06-30T20:10:09Z"
 }
 }
 }
 }
]
}
```

```
 "sessionIssuer": {
 "type": "Role",
 "principalId": "AIDAJ45Q7YFFAREEXAMPLE",
 "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"
}
]
```

## Uso de Amazon Rekognition con endpoints de Amazon VPC

Si utiliza Amazon Virtual Private Cloud (Amazon VPC) para alojar sus recursos de AWS de, 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 enlace de la 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 enlace de la VPC de tipo interfaz para Amazon Rekognition. Un punto de enlace de interfaz es una interfaz de red elástica con una dirección IP privada que actúa como punto de entrada para el tráfico dirigido a un servicio de AWS compatible. El punto de enlace ofrece conectividad escalable de confianza con Amazon Rekognition y no requiere una gateway de Internet, una instancia (NAT) de traducción de dirección de red o una conexión de VPN. Para obtener más información, consulte [What Is Amazon VPC](#) (¿Qué es Amazon VPC?) en la Guía del usuario de Amazon VPC.

AWS PrivateLink habilita los puntos de enlace de la VPC de tipo interfaz. 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 PrivateLink admite todos los puntos de enlace del Estándar federal de procesamiento de información (FIPS) de Amazon Rekognition.

## Creación de puntos de enlace de Amazon VPC para Amazon Rekognition

Puede crear dos tipos de puntos de enlace de Amazon VPC para utilizarlos con Amazon Rekognition.

- Un punto de enlace de la VPC se utilizará con las operaciones de Amazon Rekognition. Para la mayoría de los usuarios, este es el tipo más adecuado.
- Un punto de enlace de la VPC para las operaciones de Amazon Rekognition con puntos de enlace 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 su VPC, utilice la consola de Amazon VPC para crear un punto de enlace de la VPC de tipo interfaz para Amazon Rekognition. Para obtener instrucciones, consulte el procedimiento "Para crear un punto de enlace de interfaz para un servicio de AWS utilizando la consola" en [Creación de un punto de enlace de interfaz](#). Tenga en cuenta los siguientes pasos del procedimiento:

- Paso 3: ParaCategoría de servicio, eligeServicios de AWS.
- Paso 4: ParaNombre del servicio, elija una de las siguientes opciones:
  - com.amazonaws.region.rekognition: crea un punto de enlace de la VPC para las operaciones de Amazon Rekognition.
  - com.amazonaws.region.rekognition-fips: crea un punto de enlace de la VPC para las operaciones de Amazon Rekognition con puntos de enlace 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 enlace de la VPC para Amazon Rekognition

Puede crear una política para los puntos de enlace de Amazon VPC correspondiente a Amazon Rekognition y especificar lo siguiente:

- La entidad de seguridad 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 enlace de la 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 enlace de la VPC llamar al `detectFaces`Operación API. La política impide que los usuarios realicen otras operaciones de la API de Amazon Rekognition a través del punto de enlace de la VPC.

Los usuarios todavía 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 \(p. 486\)](#).

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

Modificación de la política de punto de enlace de la VPC correspondiente a 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 enlace de Amazon Rekognition elija Creación de un punto de enlace. A continuación, seleccione com.amazonaws.**Region**.rekognition y elija Create endpoint (Crear punto de enlace).
3. En el panel de navegación, elija Endpoints (Puntos de enlace).
4. Seleccione el punto de enlace com.amazonaws.**región**.rekognition y elija la pestaña Policy (Política) en la mitad inferior de la pantalla.
5. Elija Edit Policy (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 distintos AWS Programas de conformidad. 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 en relación con la conformidad al utilizar Amazon Rekognition depende de la confidencialidad de los datos, los objetivos de conformidad de su empresa y de la legislación y los reglamentos aplicables. 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.

- [AWS Recursos 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](#): 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 comutación 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 ayudan 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](#).

## Seguridad de la infraestructura en Amazon Rekognition

Al tratarse de un servicio administrado, Amazon Rekognition está protegido por los procedimientos de seguridad de red globales de que se describen en el [Amazon Web Services: Información general sobre procesos de seguridad](#) documento técnico.

Usa AWS publicadas en las llamadas a la API publicadas para obtener acceso a Amazon Rekognition a través de Los clientes deben ser compatibles con Transport Layer Security (TLS) 1.0 o una versión posterior. Recomendamos TLS 1.2 o una versión posterior. Los clientes también deben ser compatibles con conjuntos de cifrado con confidencialidad directa total (PFS) tales como Ephemeral Diffie-Hellman (DHE) o Elliptic Curve Ephemeral Diffie-Hellman (ECDHE). La mayoría de los sistemas modernos como Java 7 y posteriores son compatibles con estos modos.

Además, las solicitudes deben estar firmadas mediante un ID de clave de acceso y una clave de acceso secreta que esté asociada a una entidad de seguridad de IAM. También puede utilizar [AWS Security Token Service](#) (AWS STS) para generar credenciales de seguridad temporales para firmar solicitudes.

# Referencia de la API

Esta sección proporciona documentación sobre las operaciones de la API de Amazon Rekognition.

## Amazon Rekognition Image

- the section called “CompareFaces” (p. 515)
- the section called “CreateCollection” (p. 522)
- the section called “DeleteCollection” (p. 541)
- the section called “DeleteFaces” (p. 545)
- the section called “DescribeCollection” (p. 556)
- the section called “DetectFaces” (p. 578)
- the section called “DetectLabels” (p. 583)
- the section called “DetectModerationLabels” (p. 588)
- the section called “DetectProtectiveEquipment” (p. 592)
- the section called “DetectText” (p. 596)
- the section called “GetCelebrityInfo” (p. 602)
- the section called “IndexFaces” (p. 644)
- the section called “ListCollections” (p. 653)
- the section called “ListFaces” (p. 663)
- the section called “RecognizeCelebrities” (p. 671)
- the section called “SearchFaces” (p. 676)
- the section called “SearchFacesByImage” (p. 680)

## Etiquetas personalizadas de Amazon Rekognition

- the section called “CreateDataset” (p. 525)
- the section called “CreateProject” (p. 529)
- the section called “CreateProjectVersion” (p. 532)
- the section called “DeleteDataset” (p. 543)
- the section called “DeleteProject” (p. 548)
- the section called “DeleteProjectVersion” (p. 551)
- the section called “DescribeDataset” (p. 559)
- the section called “DescribeProjects” (p. 561)
- the section called “DescribeProjectVersions” (p. 564)
- the section called “DetectCustomLabelLabels” (p. 573)
- the section called “DistributeDatasetEntries” (p. 600)
- the section called “ListDatasetEntries” (p. 656)
- the section called “ListDatasetLabels” (p. 660)
- the section called “StartProjectVersion” (p. 709)
- the section called “StopProjectVersion” (p. 722)

- the section called “UpdateDatasetEntries” (p. 730)

## Vídeo almacenado de Amazon Rekognition

- the section called “GetCelebrityRecognition” (p. 605)
- the section called “GetContentModeration” (p. 611)
- the section called “GetFaceDetection” (p. 615)
- the section called “GetFaceSearch” (p. 620)
- the section called “GetLabelDetection” (p. 626)
- the section called “GetPersonTracking” (p. 630)
- the section called “GetSegmentDetection” (p. 635)
- the section called “GetTextDetection” (p. 640)
- the section called “StartCelebrityRecognition” (p. 685)
- the section called “StartContentModeration” (p. 689)
- the section called “StartFaceDetection” (p. 693)
- the section called “StartFaceSearch” (p. 697)
- the section called “StartLabelDetection” (p. 701)
- the section called “StartPersonTracking” (p. 705)
- the section called “StartSegmentDetection” (p. 712)
- the section called “StartTextDetection” (p. 718)

## Amazon Rekognition Video Streaming de vídeo

- the section called “CreateStreamProcessor” (p. 537)
- the section called “DeleteStreamProcessor” (p. 554)
- the section called “DescribeStreamProcessor” (p. 569)
- the section called “ListStreamProcessors” (p. 666)
- the section called “StartStreamProcessor” (p. 716)
- the section called “StopStreamProcessor” (p. 724)

### Temas

- [Acciones \(p. 513\)](#)
- [Tipos de datos \(p. 732\)](#)

## Encabezados HTTP

En este tema se describen los encabezados HTTP necesarios para llamar a una operación de Amazon Rekognition con una solicitud HTTP. Para obtener más información, consulte las [API de AWS](#).

### Encabezados HTTP

Además de los encabezados HTTP habituales, las operaciones HTTP de Amazon Rekognition tienen los siguientes encabezados obligatorios:

Encabezado	Valor	Descripción
Content-Type:	application/x-amz-json-1.1	Especifica que el contenido de la solicitud es JSON. También especifica la versión de JSON.
X-Amz-Date:	<Fecha>	La fecha que se utiliza para crear la firma en el encabezado de autorización. El formato debe ser ISO 8601 básico con el formato AAAAMMDD'T'HHMMSS'Z'. Por ejemplo, la siguiente fecha/hora 20141123T120000Z es un valor válido de x-amz-date para Amazon Rekognition.
X-Amz-Target:	RekognitionService. <operación>	La operación de Amazon Rekognition de destino. Por ejemplo, utilice RekognitionService.ListCollections para llamar a la operación ListCollections.

## Acciones

Se admiten las siguientes acciones:

- [CompareFaces \(p. 515\)](#)
- [CreateCollection \(p. 522\)](#)
- [CreateDataset \(p. 525\)](#)
- [CreateProject \(p. 529\)](#)
- [CreateProjectVersion \(p. 532\)](#)
- [CreateStreamProcessor \(p. 537\)](#)
- [DeleteCollection \(p. 541\)](#)
- [DeleteDataset \(p. 543\)](#)
- [DeleteFaces \(p. 545\)](#)
- [DeleteProject \(p. 548\)](#)
- [DeleteProjectVersion \(p. 551\)](#)
- [DeleteStreamProcessor \(p. 554\)](#)
- [DescribeCollection \(p. 556\)](#)
- [DescribeDataset \(p. 559\)](#)
- [DescribeProjects \(p. 561\)](#)
- [DescribeProjectVersions \(p. 564\)](#)
- [DescribeStreamProcessor \(p. 569\)](#)
- [DetectCustomLabels \(p. 573\)](#)
- [DetectFaces \(p. 578\)](#)
- [DetectLabels \(p. 583\)](#)
- [DetectModerationLabels \(p. 588\)](#)
- [DetectProtectiveEquipment \(p. 592\)](#)

- [DetectText \(p. 596\)](#)
- [DistributeDatasetEntries \(p. 600\)](#)
- [GetCelebrityInfo \(p. 602\)](#)
- [GetCelebrityRecognition \(p. 605\)](#)
- [GetContentModeration \(p. 611\)](#)
- [GetFaceDetection \(p. 615\)](#)
- [GetFaceSearch \(p. 620\)](#)
- [GetLabelDetection \(p. 626\)](#)
- [GetPersonTracking \(p. 630\)](#)
- [GetSegmentDetection \(p. 635\)](#)
- [GetTextDetection \(p. 640\)](#)
- [IndexFaces \(p. 644\)](#)
- [ListCollections \(p. 653\)](#)
- [ListDatasetEntries \(p. 656\)](#)
- [ListDatasetLabels \(p. 660\)](#)
- [ListFaces \(p. 663\)](#)
- [ListStreamProcessors \(p. 666\)](#)
- [ListTagsForResource \(p. 669\)](#)
- [RecognizeCelebrities \(p. 671\)](#)
- [SearchFaces \(p. 676\)](#)
- [SearchFacesByImage \(p. 680\)](#)
- [StartCelebrityRecognition \(p. 685\)](#)
- [StartContentModeration \(p. 689\)](#)
- [StartFaceDetection \(p. 693\)](#)
- [StartFaceSearch \(p. 697\)](#)
- [StartLabelDetection \(p. 701\)](#)
- [StartPersonTracking \(p. 705\)](#)
- [StartProjectVersion \(p. 709\)](#)
- [StartSegmentDetection \(p. 712\)](#)
- [StartStreamProcessor \(p. 716\)](#)
- [StartTextDetection \(p. 718\)](#)
- [StopProjectVersion \(p. 722\)](#)
- [StopStreamProcessor \(p. 724\)](#)
- [TagResource \(p. 726\)](#)
- [UntagResource \(p. 728\)](#)
- [UpdateDatasetEntries \(p. 730\)](#)

## CompareFaces

Compara un rostro en elorigenimagen de entrada con cada una de las 100 caras más grandes detectadas en eltargetimagen de entrada.

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

### Note

CompareFaces utiliza algoritmos de aprendizaje automático, que son probabilísticos. Un falso negativo es una predicción incorrecta de que una cara de la imagen de destino tiene una puntuación de confianza de similitud baja en comparación con la cara de la imagen de origen. Para reducir la probabilidad de falsos negativos, recomendamos comparar la imagen de destino con varias imágenes de origen. Si planea utilizar CompareFaces para tomar una decisión que afecte a los derechos, la privacidad o el acceso a los servicios de una persona, te recomendamos que pases el resultado a un humano para que lo revise y lo valide antes de tomar medidas.

Las imágenes de entrada y destino se pasan como bytes de imagen codificados en base 64 o como referencias a imágenes en un bucket de Amazon S3. Si utiliza la CLI de AWS para llamar a las operaciones de Amazon Rekognition, no es posible pasar bytes de imágenes. La imagen debe estar formateada como archivo PNG o JPEG.

Como respuesta, la operación devuelve una matriz de los rostros coincidentes ordenados por puntuación de similitud en orden descendente. Para cada coincidencia de cara, la respuesta proporciona un cuadro delimitador de la cara, puntos de referencia faciales, detalles de la pose (tono, balanceo y guiñada), calidad (brillo y nitidez) y valor de confianza (que indica el nivel de confianza que el cuadro delimitador contiene una cara). La respuesta también proporciona una puntuación de similitud, que indica qué tan cerca coinciden las caras.

### Note

De forma predeterminada, en la respuesta solo se devuelven caras con una puntuación de similitud superior o igual al 80%. Puede cambiar este valor especificando la `SimilarityThreshold` parámetro.

CompareFace también devuelve un conjunto de caras que no coinciden con la imagen de origen. Para cada cara, devuelve un cuadro delimitador, un valor de confianza, puntos de referencia, detalles de pose y calidad. La respuesta también devuelve información sobre el rostro de la imagen de origen, incluido el cuadro delimitador de la rostro y el valor de confianza.

La `QualityFilter` el parámetro de entrada le permite filtrar las caras detectadas que no cumplen con la barra de calidad requerida. La barra de calidad se basa en una variedad de casos de uso comunes. Usar `QualityFilter` para configurar la barra de calidad especificando `LOW`, `MEDIUM`, o bien `HIGH`. Si no desea filtrar las caras detectadas, especifique `NONE`. El valor predeterminado es `NONE`.

Si la imagen no contiene metadatos Exif, CompareFaces devuelve la información de orientación de las imágenes de origen y destino. Utilice estos valores para mostrar las imágenes con la orientación correcta de la imagen.

Si no se detectan caras en las imágenes de origen o de destino, CompareFaces devuelve un `InvalidParameterException`.

### Note

Se trata de una operación de API sin estado. Es decir, los datos devueltos por esta operación no persisten.

Para ver un ejemplo, consulte [Comparación de rostros en imágenes \(p. 163\)](#).

Esta operación requiere permisos para realizar la acción `rekognition:CompareFaces`.

## Sintaxis de la solicitud

```
{
 "QualityFilter": "string",
 "SimilarityThreshold": number,
 "SourceImage": {
 "Bytes": blob,
 "S3Object": {
 "Bucket": "string",
 "Name": "string",
 "Version": "string"
 }
 },
 "TargetImage": {
 "Bytes": blob,
 "S3Object": {
 "Bucket": "string",
 "Name": "string",
 "Version": "string"
 }
 }
}
```

## Parámetros de solicitud

La solicitud acepta los siguientes datos en formato JSON.

### QualityFilter (p. 516)

Filtro que especifica una barra de calidad para determinar cuánto filtrado se realiza para identificar caras. Las caras filtradas no se comparan. Si especifica AUTO, Amazon Rekognition elige la barra de calidad. Si especifica LOW, MEDIUM, o bien HIGH, el filtrado elimina todas las caras que no cumplen con la barra de calidad elegida. La barra de calidad se basa en una variedad de casos de uso comunes. Las detecciones de baja calidad pueden ocurrir por diversas razones. Algunos ejemplos son un objeto que se identifica mal como cara, cara demasiado borrosa o cara con una pose demasiado extrema para usarla. Si especifica NONE, no se realiza ningún filtrado. El valor predeterminado es NONE.

Para utilizar el filtrado según la calidad, la colección que está utilizando debe estar asociada a la versión 3 del modelo de rostros o superior.

Type: Cadena

Valores válidos: NONE | AUTO | LOW | MEDIUM | HIGH

Obligatorio: No

### SimilarityThreshold (p. 516)

El nivel mínimo de confianza en la cara coincide con el que debe cumplir una partida para incluirse en el FaceMatches matriz.

Type: Float

Rango válido: Valor mínimo de 0. Valor máximo de 100.

Obligatorio: No

### SourceImage (p. 516)

La imagen de entrada como bytes codificados en base64 o un objeto S3. Si utiliza la CLI de AWS para llamar a las operaciones de Amazon Rekognition, no es posible pasar bytes de imágenes codificados en base64.

Si utiliza un SDK de AWS para llamar a Amazon Rekognition, es posible que no tenga que codificar en base 64 bytes de imagen pasados mediante elBytes. Para obtener más información, consulte [Especificaciones de imágenes \(p. 29\)](#).

Tipo: objeto [Image \(p. 786\)](#)

Obligatorio: Sí

[TargetImage \(p. 516\)](#)

La imagen de destino como bytes codificados en base64 o un objeto S3. Si utiliza la CLI de AWS para llamar a las operaciones de Amazon Rekognition, no es posible pasar bytes de imágenes codificados en base64.

Si utiliza un SDK de AWS para llamar a Amazon Rekognition, es posible que no tenga que codificar en base 64 bytes de imagen pasados mediante elBytes. Para obtener más información, consulte [Especificaciones de imágenes \(p. 29\)](#).

Tipo: objeto [Image \(p. 786\)](#)

Obligatorio: Sí

## Sintaxis de la respuesta

```
{
 "FaceMatches": [
 {
 "Face": {
 "BoundingBox": {
 "Height": number,
 "Left": number,
 "Top": number,
 "Width": number
 },
 "Confidence": number,
 "Emotions": [
 {
 "Confidence": number,
 "Type": "string"
 }
],
 "Landmarks": [
 {
 "Type": "string",
 "X": number,
 "Y": number
 }
],
 "Pose": {
 "Pitch": number,
 "Roll": number,
 "Yaw": number
 },
 "Quality": {
 "Brightness": number,
 "Sharpness": number
 },
 "Smile": {
 "Confidence": number,
 "Value": boolean
 }
 },
 "Similarity": number
 }
]
}
```

```
 },
],
 "SourceImageFace": {
 "BoundingBox": {
 "Height": number,
 "Left": number,
 "Top": number,
 "Width": number
 },
 "Confidence": number
 },
 "SourceImageOrientationCorrection": "string",
 "TargetImageOrientationCorrection": "string",
 "UnmatchedFaces": [
 {
 "BoundingBox": {
 "Height": number,
 "Left": number,
 "Top": number,
 "Width": number
 },
 "Confidence": number,
 "Emotions": [
 {
 "Confidence": number,
 "Type": "string"
 }
],
 "Landmarks": [
 {
 "Type": "string",
 "X": number,
 "Y": number
 }
],
 "Pose": {
 "Pitch": number,
 "Roll": number,
 "Yaw": number
 },
 "Quality": {
 "Brightness": number,
 "Sharpness": number
 },
 "Smile": {
 "Confidence": number,
 "Value": boolean
 }
 }
]
}
```

## Elementos de respuesta

Si la acción se realiza correctamente, el servicio devuelve una respuesta HTTP 200.

El servicio devuelve los datos siguientes en formato JSON.

### FaceMatches (p. 517)

Matriz de rostros de la imagen de destino que coincidan con el rostro de la imagen de origen. Cada `CompareFacesMatch` objeto proporciona el cuadro delimitador, el nivel de confianza que el cuadro delimitador contiene una cara y la puntuación de similitud de la cara en el cuadro delimitador y la cara de la imagen de origen.

Type: Matriz de[CompareFacesMatch \(p. 750\)](#)objects

[SourceImageFace \(p. 517\)](#)

La cara de la imagen de origen que se utilizó para la comparación.

Tipo: objeto [ComparedSourceImageFace \(p. 749\)](#)

[SourceImageOrientationCorrection \(p. 517\)](#)

El valor de[SourceImageOrientationCorrection](#)siempre es nulo.

Si la imagen de entrada está en formato .jpeg, podría contener metadatos de formato de archivo de imagen intercambiable (EXIF) que incluya la orientación de la imagen. Amazon Rekognition utiliza esta información de orientación para realizar la corrección de imagen. Las coordenadas del cuadro delimitador se traducen para representar las ubicaciones de los objetos después de utilizar la información de orientación de los metadatos Exif para corregir la orientación de la imagen. Las imágenes en formato .png no contienen metadatos Exif.

Amazon Rekognition no realiza la corrección de imagen para imágenes en formato.png ni imágenes.jpeg sin información de orientación en los metadatos Exif de la imagen. Las coordenadas del cuadro delimitador no se traducen y representan las ubicaciones de los objetos antes de girar la imagen.

Type: Cadena

Valores válidos: ROTATE\_0 | ROTATE\_90 | ROTATE\_180 | ROTATE\_270

[TargetImageOrientationCorrection \(p. 517\)](#)

El valor de[TargetImageOrientationCorrection](#)siempre es nulo.

Si la imagen de entrada está en formato .jpeg, podría contener metadatos de formato de archivo de imagen intercambiable (EXIF) que incluya la orientación de la imagen. Amazon Rekognition utiliza esta información de orientación para realizar la corrección de imagen. Las coordenadas del cuadro delimitador se traducen para representar las ubicaciones de los objetos después de utilizar la información de orientación de los metadatos Exif para corregir la orientación de la imagen. Las imágenes en formato .png no contienen metadatos Exif.

Amazon Rekognition no realiza la corrección de imagen para imágenes en formato.png ni imágenes.jpeg sin información de orientación en los metadatos Exif de la imagen. Las coordenadas del cuadro delimitador no se traducen y representan las ubicaciones de los objetos antes de girar la imagen.

Type: Cadena

Valores válidos: ROTATE\_0 | ROTATE\_90 | ROTATE\_180 | ROTATE\_270

[UnmatchedFaces \(p. 517\)](#)

Matriz de rostros de la imagen de destino que no coincidan con el rostro de la imagen de origen.

Type: Matriz de[ComparedFace \(p. 747\)](#)objects

## Errores

[AccessDeniedException](#)

No tiene autorización para realizar la acción.

Código de estado HTTP: 400

#### ImageTooLargeException

La imagen de entrada tamaño supera el límite permitido. Si estás llamando [DetectProtectiveEquipment \(p. 592\)](#), el tamaño o la resolución de la imagen supera el límite permitido. Para obtener más información, consulte [Directrices y cuotas de Amazon Rekognition \(p. 847\)](#).

Código de estado HTTP: 400

#### InternalServerError

Amazon Lex ha tenido un problema de servicio. Pruebe la llamada de nuevo.

Código de estado HTTP: 500

#### InvalidImageFormatException

No se admite el formato de imagen proporcionado.

Código de estado HTTP: 400

#### InvalidParameterException

El parámetro de entrada infringió una restricción. Valide el parámetro antes de llamar a la operación de la API de nuevo.

Código de estado HTTP: 400

#### InvalidS3ObjectException

Amazon Rekognition no puede obtener acceso al objeto de S3 especificado en la solicitud.

Código de estado HTTP: 400

#### ProvisionedThroughputExceededException

El número de solicitudes ha superado su límite de rendimiento. Si necesita aumentar este límite, póngase en contacto con Amazon Rekognition.

Código de estado HTTP: 400

#### ThrottlingException

Amazon Lex no puede procesar temporalmente la solicitud. Pruebe la llamada de nuevo.

Código de estado HTTP: 500

## Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- [AWS Command Line Interface](#)
- [SDK de AWS para .NET](#)
- [AWS SDK para C++](#)
- [AWS SDK para Go](#)
- [AWSSDK para Java V2](#)
- [AWS SDK para JavaScript](#)
- [SDK de AWS para PHP V3](#)
- [SDK de AWS para Python](#)
- [SDK de AWS para Ruby V3](#)



## CreateCollection

Crea una colección en una región de AWS. Puede añadir rostros a la colección mediante [laIndexFaces \(p. 644\)](#).

Por ejemplo, puede haber creado colecciones, una para cada usuario de la aplicación de. A continuación, un usuario puede indexar caras mediante el `IndexFaces` la operación y la persistencia da como resultado una colección específica. A continuación, un usuario puede buscar caras en la colección en el contenedor específico del usuario.

Al crear una colección, se asocia a la última versión de la versión de modelo facial.

### Note

Los nombres de colección distinguen entre mayúsculas y min

Esta operación requiere permisos para realizar la acción `rekognition:CreateCollection`. Si quieres etiquetar tu colección, también necesitas permiso para realizar el `rekognition:TagResource`.

## Sintaxis de la solicitud

```
{
 "CollectionId": "string",
 "Tags": {
 "string" : "string"
 }
}
```

## Parámetros de solicitud

La solicitud acepta los siguientes datos en formato JSON.

### [CollectionId \(p. 522\)](#)

ID de la colección que está creando.

Type: Cadena

Limitaciones de longitud: Longitud mínima de 1. La longitud máxima es de 255 caracteres.

Patrón: [a-zA-Z0-9\_.\-\+]

Obligatorio: Sí

### [Tags \(p. 522\)](#)

Conjunto de etiquetas (pares clave-valor) que deseé asociar a la colección de.

Type: Asignación de cadena a cadena

Entradas de mapa: El número mínimo es 0 elementos. Número máximo de 200 elementos.

Limitaciones de longitud de clave: Longitud mínima de 1. La longitud máxima es de 128 caracteres.

Patrón de clave: ^(?!aws:)[\p{L}\p{Z}\p{N}\_.:/=+\-\@]\*\$

Restricciones de longitud de valor: Longitud mínima de 0. La longitud máxima es de 256 caracteres.

Patrón de valor: ^([\p{L}\p{Z}\p{N}\_.:/=+\-\@]\*)\$

Obligatorio: No

## Sintaxis de la respuesta

```
{
 "CollectionArn": "string",
 "FaceModelVersion": "string",
 "StatusCode": number
}
```

## Elementos de respuesta

Si la acción se realiza correctamente, el servicio devuelve una respuesta HTTP 200.

El servicio devuelve los datos siguientes en formato JSON.

### [CollectionArn \(p. 523\)](#)

Nombre de recurso de Amazon (ARN) de la colección. Puede usarlo para administrar los permisos de sus recursos.

Type: Cadena

### [FaceModelVersion \(p. 523\)](#)

Número de versión del modelo de detección de rostros asociado a la colección que está creando.

Type: Cadena

### [StatusCode \(p. 523\)](#)

Código de estado HTTP que indica el resultado de la operación.

Type: Entero

Rango válido: Valor mínimo de 0.

## Errores

### [AccessDeniedException](#)

No tiene autorización para realizar la acción.

Código de estado HTTP: 400

### [InternalServerError](#)

Amazon Lex ha tenido un problema de servicio. Pruebe la llamada de nuevo.

Código de estado HTTP: 500

### [InvalidArgumentException](#)

El parámetro de entrada infringió una restricción. Valide el parámetro antes de llamar a la operación de la API de nuevo.

Código de estado HTTP: 400

### [ProvisionedThroughputExceededException](#)

El número de solicitudes ha superado su límite de rendimiento. Si necesita aumentar este límite, póngase en contacto con Amazon Rekognition.

Código de estado HTTP: 400

#### ResourceAlreadyExistsException

Ya existe un recurso con el ID especificado.

Código de estado HTTP: 400

#### ServiceQuotaExceededException

El tamaño del recurso supera el límite permitido. Para obtener más información, consulte [Directrices y cuotas de Amazon Rekognition \(p. 847\)](#).

Código de estado HTTP: 400

#### ThrottlingException

Amazon Lex no puede procesar temporalmente la solicitud. Pruebe la llamada de nuevo.

Código de estado HTTP: 500

## Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- [AWS Command Line Interface](#)
- [SDK de AWS para .NET](#)
- [AWS SDK para C++](#)
- [AWS SDK para Go](#)
- [AWSSDK para Java V2](#)
- [AWS SDK para JavaScript](#)
- [SDK de AWS para PHP V3](#)
- [SDK de AWS para Python](#)
- [SDK de AWS para Ruby V3](#)

## CreateDataset

Crea un nuevo conjunto de datos de etiquetas personalizadas de Amazon Rekognition. Puede crear un conjunto de datos mediante un archivo de manifiesto de formato de Amazon Sagemaker o copiando un conjunto de datos de etiquetas personalizadas de Amazon Rekognition existente.

Para crear un conjunto de datos de formación para un proyecto, especifique `train` por el valor `deDatasetType`. Para crear el conjunto de datos de prueba de un proyecto, especifique `test` por el valor `deDatasetType`.

La respuesta de `CreateDataset` es el nombre de recurso de Amazon (ARN) del conjunto de datos de. La creación de un conjunto de datos tarda un tiempo en completarse.

Usar [DescribeDataset \(p. 559\)](#) para comprobar el estado actual. El conjunto de datos creado correctamente si el valor de `statuses` es `CREATE_COMPLETE`.

Para comprobar si se ha producido algún error no terminal, llame [ListDatasetEntries \(p. 656\)](#) y comprobar la presencia de `errors` en las líneas JSON.

La creación de conjuntos de datos falla si se produce un error de terminal (`Status=CREATE_FAILED`). En la actualidad, no se puede obtener acceso a la información de error de la terminal.

Para obtener más información, consulte [Creación de conjuntos de datos](#).

Esta operación requiere permisos para realizar la acción `rekognition:CreateDataset`. Si desea copiar un conjunto de datos existente, también necesita permiso para realizar el `rekognition>ListDatasetEntries` acción.

## Sintaxis de la solicitud

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

## Parámetros de solicitud

La solicitud acepta los siguientes datos en formato JSON.

### DatasetSource (p. 525)

Los archivos de origen del conjunto de datos. Puede especificar el ARN de un conjunto de datos existente o especificar la ubicación del bucket de Amazon S3 de un archivo de manifiesto con formato de Amazon Sagemaker. Si no especifica `datasetSource`, se crea un conjunto de datos vacío. Para añadir imágenes etiquetadas al conjunto de datos, puede utilizar la consola de o llamar a [UpdateDatasetEntries \(p. 730\)](#).

Tipo: objeto [DatasetSource \(p. 761\)](#)

Obligatorio: No

### [DatasetType \(p. 525\)](#)

Tipo de conjunto de datos. Especifique `.train`para crear un conjunto de datos de formación. Especifique `.test`para crear un conjunto de datos de prueba.

Type: Cadena

Valores válidos: `TRAIN` | `TEST`

Obligatorio: Sí

### [ProjectArn \(p. 525\)](#)

El ARN del proyecto de etiquetas personalizadas de Amazon Rekognition al que desea asignar el conjunto de datos de.

Type: Cadena

Restricciones de longitud: Longitud mínima de 20. La longitud máxima es de 2048 caracteres.

Patrón: (^arn:[a-zA-Z\d-]+:rekognition:[a-zA-Z\d-]+:\d{12}:project\/[a-zA-Z0-9\_.\-\-]{1,255}\/[0-9]+\$)

Obligatorio: Sí

## Sintaxis de la respuesta

```
{
 "DatasetArn": "string"
}
```

## Elementos de respuesta

Si la acción se realiza correctamente, el servicio devuelve una respuesta HTTP 200.

El servicio devuelve los datos siguientes en formato JSON.

### [DatasetArn \(p. 526\)](#)

El ARN del conjunto de datos creado de etiquetas personalizadas de Amazon Rekognition.

Type: Cadena

Restricciones de longitud: Longitud mínima de 20. La longitud máxima es de 2048 caracteres.

Patrón: (^arn:[a-zA-Z\d-]+:rekognition:[a-zA-Z\d-]+:\d{12}:project\/[a-zA-Z0-9\_.\-\-]{1,255}\/dataset\/(train|test)\/[0-9]+\$)

## Errores

### AccessDeniedException

No tiene autorización para realizar la acción.

Código de estado HTTP: 400

### InternalServerError

Amazon Lex ha tenido un problema de servicio. Pruebe la llamada de nuevo.

Código de estado HTTP: 500

InvalidParameterException

El parámetro de entrada infringió una restricción. Valide el parámetro antes de llamar a la operación de la API de nuevo.

Código de estado HTTP: 400

InvalidS3ObjectException

Amazon Rekognition no puede obtener acceso al objeto de S3 especificado en la solicitud.

Código de estado HTTP: 400

LimitExceededException

Un límite de servicio de Amazon Rekognition se ha superado. Por ejemplo, si inicia demasiados trabajos de Amazon Rekognition Video simultáneamente, llama para iniciar operaciones (`StartLabelDetection`, por ejemplo) elevará un `LimitExceededException` excepción (código de estado HTTP: 400) hasta que el número de trabajos ejecutados simultáneamente se encuentre por debajo del límite de servicio de Amazon Rekognition.

Código de estado HTTP: 400

ProvisionedThroughputExceededException

El número de solicitudes ha superado su límite de rendimiento. Si necesita aumentar este límite, póngase en contacto con Amazon Rekognition.

Código de estado HTTP: 400

ResourceAlreadyExistsException

Ya existe un recurso con el ID especificado.

Código de estado HTTP: 400

ResourceNotFoundException

No se encuentra el recurso especificado en la solicitud.

Código de estado HTTP: 400

ThrottlingException

Amazon Lex no puede procesar temporalmente la solicitud. Pruebe la llamada de nuevo.

Código de estado HTTP: 500

## Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- [AWS Command Line Interface](#)
- [SDK de AWS para .NET](#)
- [AWS SDK para C++](#)
- [AWS SDK para Go](#)
- [AWSSDK para Java V2](#)
- [AWS SDK para JavaScript](#)
- [SDK de AWS para PHP V3](#)

- [SDK de AWS para Python](#)
- [SDK de AWS para Ruby V3](#)

## CreateProject

Crea un nuevo proyecto de etiquetas personalizadas de Amazon Rekognition. Un proyecto es un grupo de recursos (conjuntos de datos, versiones de modelos) que se utilizan para crear y administrar modelos de etiquetas personalizadas de Amazon Rekognition.

Esta operación requiere permisos para realizar la acción `rekognition:CreateProject`.

### Sintaxis de la solicitud

```
{
 "ProjectName": "string"
}
```

### Parámetros de solicitud

La solicitud acepta los siguientes datos en formato JSON.

#### [ProjectName \(p. 529\)](#)

Nombre del proyecto que se va a crear.

Type: Cadena

Restricciones de longitud: Longitud mínima de 1. La longitud máxima es de 255 caracteres.

Patrón: [a-zA-Z0-9\_.\-\-]+

Obligatorio: Sí

### Sintaxis de la respuesta

```
{
 "ProjectArn": "string"
}
```

### Elementos de respuesta

Si la acción se realiza correctamente, el servicio devuelve una respuesta HTTP 200.

El servicio devuelve los datos siguientes en formato JSON.

#### [ProjectArn \(p. 529\)](#)

El nombre de recurso de Amazon (ARN) del nuevo proyecto. Puede utilizar el ARN para configurar el acceso de IAM al proyecto.

Type: Cadena

Restricciones de longitud: Longitud mínima de 20. La longitud máxima es de 2048 caracteres.

Patrón: (^arn:[a-zA-Z0-9\_.-]+:rekognition:[a-zA-Z0-9\_.-]+\d{12}:project\/[a-zA-Z0-9\_.-]{1,255}\/[0-9]+\$)

## Errores

### AccessDeniedException

No tiene autorización para realizar la acción.

Código de estado HTTP: 400

### InternalServerError

Amazon Lex ha tenido un problema de servicio. Pruebe la llamada de nuevo.

Código de estado HTTP: 500

### InvalidParameterException

El parámetro de entrada infringió una restricción. Valide el parámetro antes de llamar a la operación de la API de nuevo.

Código de estado HTTP: 400

### LimitExceedededException

Se ha superado un límite de servicio de Amazon Rekognition. Por ejemplo, si inicia demasiados trabajos de Amazon Rekognition Video simultáneamente, llama para iniciar operaciones (`StartLabelDetection`, por ejemplo) elevará un `LimitExceedededException` excepción (código de estado HTTP: 400) hasta que el número de trabajos ejecutados simultáneamente se encuentre por debajo del límite de servicio de Amazon Rekognition.

Código de estado HTTP: 400

### ProvisionedThroughputExceededException

El número de solicitudes ha superado su límite de rendimiento. Si necesita aumentar este límite, póngase en contacto con Amazon Rekognition.

Código de estado HTTP: 400

### ResourceInUseException

El recurso especificado ya se está utilizando.

Código de estado HTTP: 400

### ThrottlingException

Amazon Lex no puede procesar temporalmente la solicitud. Pruebe la llamada de nuevo.

Código de estado HTTP: 500

## Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- [AWS Command Line Interface](#)
- [SDK de AWS para .NET](#)
- [AWS SDK para C++](#)
- [AWS SDK para Go](#)
- [AWSSDK para Java V2](#)
- [AWS SDK para JavaScript](#)

- [SDK de AWS para PHP V3](#)
- [SDK de AWS para Python](#)
- [SDK de AWS para Ruby V3](#)

## CreateProjectVersion

Crea una versión nueva de un modelo y comienza el entrenamiento. Los modelos se administran como parte de un proyecto de etiquetas personalizadas de Amazon Rekognition. La respuesta de `CreateProjectVersion` incluye un Nombre de recurso de Amazon (ARN) para la versión del modelo.

La formación utiliza los conjuntos de datos de formación y prueba asociados al proyecto. Para obtener más información, consulte [Creación de conjuntos de datos de formación y pruebas](#).

### Note

Puede entrenar un modelo en un proyecto que no tiene conjuntos de datos asociados especificando archivos de manifiesto en `elTrainingData` y `TestingData`.

Si abre la consola después de entrenar un modelo con archivos de manifiesto, Amazon Rekognition Custom Labels crea los conjuntos de datos para usted utilizando los archivos de manifiesto más recientes. Ya no se puede entrenar una versión de modelo para el proyecto especificando archivos de manifiesto.

En lugar de entrenar con un proyecto sin conjuntos de datos asociados, le recomendamos que utilice los archivos de manifiesto para crear conjuntos de datos de formación y prueba para el proyecto.

El entrenamiento tarda un tiempo en completarse. Puede obtener el estado actual llamando `aDescribeProjectVersions` (p. 564). La formación se completó correctamente si el valor de `statusField` es `TRAINING_COMPLETED`.

Si la formación falla, consulte [Depuración de una formación de modelo fallida](#).

Una vez finalizado satisfactoriamente la formación, llame `DescribeProjectVersions` (p. 564) para obtener los resultados de la formación y evaluar el modelo. Para obtener más información, consulte [Mejora de un modelo de etiquetas personalizadas de Amazon Rekognition](#).

Después de evaluar el modelo, se inicia el modelo llamando a `StartProjectVersion` (p. 709).

Esta operación requiere permisos para realizar la acción `rekognition:CreateProjectVersion`.

## Sintaxis de la solicitud

```
{
 "KmsKeyId": "string",
 "OutputConfig": {
 "S3Bucket": "string",
 "S3KeyPrefix": "string"
 },
 "ProjectArn": "string",
 "Tags": {
 "string" : "string"
 },
 "TestingData": {
 "Assets": [
 {
 "GroundTruthManifest": {
 "S3Object": {
 "Bucket": "string",
 "Name": "string",
 "Version": "string"
 }
 }
 }
],
 "AutoCreate": boolean
 }
}
```

```
},
"TrainingData": {
 "Assets": [
 {
 "GroundTruthManifest": {
 "S3Object": {
 "Bucket": "string",
 "Name": "string",
 "Version": "string"
 }
 }
 }
],
 "VersionName": "string"
}
```

## Parámetros de solicitud

La solicitud acepta los siguientes datos en formato JSON.

### [KmsKeyId \(p. 532\)](#)

El identificador de la clave de AWS Key Management Service (clave de AWS KMS). Puede proporcionar el Nombre de recurso de Amazon (ARN) de la clave KMS, el ID de la clave de KMS, un alias para la clave de KMS o un alias. La clave se utiliza para cifrar las imágenes de formación y pruebas copiadas en el servicio para la formación de modelos. Las imágenes de origen no se ven afectadas. La clave también se utiliza para cifrar los resultados de formación y los archivos de manifiesto escritos en el bucket de Amazon S3 de salida (`OutputConfig`).

Si elige utilizar su propia clave KMS, necesita los siguientes permisos en la clave KMS.

- `kms:CreateGrant`
- `kms:DescribeKey`
- `kms:GenerateDataKey`
- `kms:Decrypt`

Si no especifica un valor para `KmsKeyId`, las imágenes copiadas en el servicio se cifran mediante una clave que AWS posee y administra.

Type: Cadena

Restricciones de longitud: Longitud mínima de 1. La longitud máxima es de 2048 caracteres.

Patrón: ^[A-Za-z0-9][A-Za-z0-9:\_/+=@.-]{0,2048}\$

Obligatorio: No

### [OutputConfig \(p. 532\)](#)

La ubicación del bucket de Amazon S3 para almacenar los resultados de la formación. El bucket de S3 puede estar en cualquier cuenta de AWS siempre que la persona que llama tenga `S3:PutObject` permisos en el bucket de S3.

Tipo: objeto [OutputConfig \(p. 800\)](#)

Obligatorio: Sí

### [ProjectArn \(p. 532\)](#)

El ARN del proyecto Etiquetas personalizadas de Amazon Rekognition que administra el modelo que desea entrenar.

Type: Cadena

Restricciones de longitud: Longitud mínima de 20. La longitud máxima es de 2048 caracteres.

Patrón: (^arn:[a-zA-Z-]+:rekognition:[a-zA-Z-]+:\d{12}:project\/[a-zA-Z0-9\_.\-\\_]{1,255}\/[0-9]+\$)

Obligatorio: Sí

#### [Tags \(p. 532\)](#)

Conjunto de etiquetas (pares clave-valor) que deseé asociar al modelo.

Type: Asignación de cadena a cadena

Entradas de mapa: El número mínimo es 0 elementos. Número máximo de 200 elementos.

Restricciones de longitud clave: Longitud mínima de 1. La longitud máxima es de 128 caracteres.

Patrón de clave: ^(? !aws :)[\p{L}\p{Z}\p{N}\_.:/=+\-@]\*\$

Restricciones de longitud de valor: Longitud mínima de 0. La longitud máxima es de 256 caracteres.

Patrón de valor: ^([\p{L}\p{Z}\p{N}\_.:/=+\-@]\*)\$

Obligatorio: No

#### [TestingData \(p. 532\)](#)

Especifica un manifiesto externo que utiliza el servicio para probar el modelo. Si especifica TestingData también debe especificar TrainingData. El proyecto no debe tener conjuntos de datos asociados.

Tipo: objeto [TestingData \(p. 835\)](#)

Obligatorio: No

#### [TrainingData \(p. 532\)](#)

Especifica un manifiesto externo que utilizan los servicios para formar el modelo. Si especifica TrainingData también debe especificar TestingData. El proyecto no debe tener conjuntos de datos asociados.

Tipo: objeto [TrainingData \(p. 840\)](#)

Obligatorio: No

#### [VersionName \(p. 532\)](#)

Un nombre para la versión del modelo. Este valor debe ser único.

Type: Cadena

Restricciones de longitud: Longitud mínima de 1. La longitud máxima es de 255 caracteres.

Patrón: [a-zA-Z0-9\_.\-\\_]+

Obligatorio: Sí

## [Sintaxis de la respuesta](#)

```
{
```

```
 "ProjectVersionArn": "string"
}
```

## Elementos de respuesta

Si la acción se realiza correctamente, el servicio devuelve una respuesta HTTP 200.

El servicio devuelve los datos siguientes en formato JSON.

### [ProjectVersionArn \(p. 534\)](#)

El ARN de la versión de modelo que se creó. Usar `DescribeProjectVersion` para obtener el estado actual de la operación de formación.

Type: Cadena

Restricciones de longitud: Longitud mínima de 20. La longitud máxima es de 2048 caracteres.

Patrón: (^arn:[a-z\d-]+:rekognition:[a-z\d-]+\:\d{12}:project\/[a-zA-Z0-9\_.\-\-]{1,255}\/version\/[a-zA-Z0-9\_.\-\-]{1,255}\/[0-9]+\$)

## Errores

### [AccessDeniedException](#)

No tiene autorización para realizar la acción.

Código de estado HTTP: 400

### [InternalServerError](#)

Amazon Lex ha tenido un problema de servicio. Pruebe la llamada de nuevo.

Código de estado HTTP: 500

### [InvalidParameterException](#)

El parámetro de entrada infringió una restricción. Valide el parámetro antes de llamar a la operación de la API de nuevo.

Código de estado HTTP: 400

### [LimitExceededException](#)

Se ha superado un límite de servicio de Amazon Rekognition. Por ejemplo, si inicia demasiados trabajos de Amazon Rekognition Video simultáneamente, llama para iniciar operaciones (`StartLabelDetection`, por ejemplo) elevará un `LimitExceededException` excepción (código de estado HTTP: 400) hasta que el número de trabajos ejecutados simultáneamente se encuentre por debajo del límite de servicio de Amazon Rekognition.

Código de estado HTTP: 400

### [ProvisionedThroughputExceededException](#)

El número de solicitudes ha superado su límite de rendimiento. Si quieras aumentar este límite, ponte en contacto con Amazon Rekognition.

Código de estado HTTP: 400

### [ResourceInUseException](#)

El recurso especificado ya se está utilizando.

Código de estado HTTP: 400  
`ResourceNotFoundException`

No se encuentra el recurso especificado en la solicitud.

Código de estado HTTP: 400  
`ServiceQuotaExceededException`

El tamaño del recurso supera el límite permitido. Para obtener más información, consulte [Directrices y cuotas de Amazon Rekognition \(p. 847\)](#).

Código de estado HTTP: 400  
`ThrottlingException`

Amazon Lex no puede procesar temporalmente la solicitud. Pruebe la llamada de nuevo.

Código de estado HTTP: 500

## Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- [AWS Command Line Interface](#)
- [SDK de AWS para .NET](#)
- [AWS SDK para C++](#)
- [AWS SDK para Go](#)
- [AWSSDK para Java V2](#)
- [AWS SDK para JavaScript](#)
- [SDK de AWS para PHP V3](#)
- [SDK de AWS para Python](#)
- [SDK de AWS para Ruby V3](#)

## CreateStreamProcessor

Crea un procesador de streaming de Amazon Rekognition que puede utilizar para detectar y reconocer rostros en un vídeo en streaming.

Amazon Rekognition Video es un consumidor de vídeo en directo de Amazon Kinesis Video Streams. Amazon Rekognition Video envía los resultados de análisis a Amazon Kinesis Data Streams.

Proporciona como entrada una transmisión de vídeo de Kinesis (`Input`) y una secuencia de datos de Kinesis (`Output`) flujo de. También se especifican los criterios de reconocimiento facial en `Settings`. Por ejemplo, la colección que contiene caras que desea reconocer. Usar `Name` para asignar un identificador para el procesador de streaming de. Usar `Name` para administrar el procesador de streaming. Por ejemplo, puede empezar a procesar el vídeo de origen llamando [StartStreamProcessor \(p. 716\)](#) con `Name`.

Una vez que haya terminado de analizar un vídeo en streaming, utilice [StopStreamProcessor \(p. 724\)](#) para detener el procesamiento. Puede eliminar el procesador de streaming llamando [DeleteStreamProcessor \(p. 554\)](#).

Esta operación requiere permisos para realizar la acción `rekognition:CreateStreamProcessor`. Si desea etiquetar el procesador de streaming, también necesita permiso para realizar el `rekognition:TagResource`.

### Sintaxis de la solicitud

```
{
 "Input": {
 "KinesisVideoStream": {
 "Arn": "string"
 }
 },
 "Name": "string",
 "Output": {
 "KinesisDataStream": {
 "Arn": "string"
 }
 },
 "RoleArn": "string",
 "Settings": {
 "FaceSearch": {
 "CollectionId": "string",
 "FaceMatchThreshold": number
 }
 },
 "Tags": {
 "string" : "string"
 }
}
```

### Parámetros de solicitud

La solicitud acepta los siguientes datos en formato JSON.

#### Input (p. 537)

Transmisión de vídeo de Kinesis que proporciona la transmisión de vídeo de origen. Si utiliza la interfaz de línea de AWS de línea de línea de línea de línea de línea de línea `deStreamProcessorInput`.

Tipo: objeto [StreamProcessorInput \(p. 829\)](#)

Obligatorio: Sí

Name (p. 537)

Identificador que se asigna al procesador de streaming. Puede usar `Name` para administrar el procesador de streaming. Por ejemplo, puede obtener el estado actual del procesador de streaming llamando a [DescribeStreamProcessor \(p. 569\)](#). `Name` es idempotente.

Type: Cadena

Restricciones de longitud: Longitud mínima de 1. La longitud máxima es de 128 caracteres.

Patrón: [a-zA-Z0-9 .\-\+]

### Obligatorio: Sí

## Output (p. 537)

Transmisión de datos de Kinesis en la que Amazon Rekognition Video pone los resultados del análisis. Si utiliza la interfaz de línea de AWS de línea de línea de línea de línea de línea de línea de StreamProcessorOutput.

Tipo: objeto StreamProcessorOutput (p. 830)

Obligatorio: Sí

## RoleArn (p. 537)

ARN de la función de IAM que permite tener acceso al procesador de streaming de.

Type: Cadena

Patrón: arn:aws:iam::\d{12}:role/?[a-zA-Z\_0-9+=,.@\\-/\_]+

## Obligatorio: Sí

## Settings (p. 537)

Parámetros de entrada de reconocimiento facial que debe utilizar el procesador de secuencias. Incluye la colección que se utilizará para el reconocimiento facial y los atributos de cara que se van a detectar.

Tipo: objeto StreamProcessorSettings (p. 831)

Obligatorio: Sí

## Tags (p. 537)

Conjunto de etiquetas (pares clave-valor) que deseé asociar al procesador de streaming de.

Type: Asignación de cadena a cadena

Entradas de mapa: El número mínimo es 0 elementos. Número máximo de 200 elementos.

Limitaciones de longitud clave: Longitud mínima de 1. La longitud máxima es de 128 caracteres.

Patrón de clave: ^(? !aws :)[\p{L}\p{Z}\p{N}\_.:/=+\-@]\*\$

Restricciones de longitud de valor: Longitud mínima de 0. La longitud máxima es de 256 caracteres.

Patrón de valor: ^([ \p{L}\p{Z}\p{N} .:/=+\-@]\*\$)

Obligatorio: No

## Sintaxis de la respuesta

1

```
 "StreamProcessorArn": "string"
}
```

## Elementos de respuesta

Si la acción se realiza correctamente, el servicio devuelve una respuesta HTTP 200.

El servicio devuelve los datos siguientes en formato JSON.

### [StreamProcessorArn \(p. 538\)](#)

ARN para el procesador de streaming recién creado.

Type: Cadena

Patrón: (^arn:[a-z\d-]+:rekognition:[a-z\d-]+\:\d{12}:streamprocessor\/.+)\$)

## Errores

### `AccessDeniedException`

No tiene autorización para realizar la acción.

Código de estado HTTP: 400

### `InternalServerError`

Amazon Lex ha tenido un problema de servicio. Pruebe la llamada de nuevo.

Código de estado HTTP: 500

### `InvalidParameterException`

El parámetro de entrada infringió una restricción. Valide el parámetro antes de llamar a la operación de la API de nuevo.

Código de estado HTTP: 400

### `LimitExceedededException`

Se ha superado un límite de servicio de Amazon Rekognition. Por ejemplo, si inicia demasiados trabajos de Amazon Rekognition Video simultáneamente, llama para iniciar operaciones (`StartLabelDetection`, por ejemplo) elevará un `LimitExceedededException` excepción (código de estado HTTP: 400) hasta que el número de trabajos ejecutados simultáneamente se encuentre por debajo del límite de servicio de Amazon Rekognition.

Código de estado HTTP: 400

### `ProvisionedThroughputExceededException`

El número de solicitudes ha superado su límite de rendimiento. Si necesita aumentar este límite, póngase en contacto con Amazon Rekognition.

Código de estado HTTP: 400

### `ResourceInUseException`

El recurso especificado ya se está utilizando.

Código de estado HTTP: 400

### `ServiceQuotaExceedededException`

El tamaño del recurso supera el límite permitido. Para obtener más información, consulte [Directrices y cuotas de Amazon Rekognition \(p. 847\)](#).

Código de estado HTTP: 400

ThrottlingException

Amazon Lex no puede procesar temporalmente la solicitud. Pruebe la llamada de nuevo.

Código de estado HTTP: 500

## Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- [AWS Command Line Interface](#)
- [SDK de AWS para .NET](#)
- [AWS SDK para C++](#)
- [AWS SDK para Go](#)
- [AWSSDK para Java V2](#)
- [AWS SDK para JavaScript](#)
- [SDK de AWS para PHP V3](#)
- [SDK de AWS para Python](#)
- [SDK de AWS para Ruby V3](#)

## DeleteCollection

Elimina la colección especificada. Tenga en cuenta que esta operación elimina todos los rostros de la colección. Para ver un ejemplo, consulte [Eliminación de una colección \(p. 201\)](#).

Esta operación requiere permisos para realizar la acción `rekognition:DeleteCollection`.

### Sintaxis de la solicitud

```
{
 "CollectionId": "string"
}
```

### Parámetros de solicitud

La solicitud acepta los siguientes datos en formato JSON.

#### [CollectionId \(p. 541\)](#)

El ID de la colección que eliminar.

Type: Cadena

Limitaciones de longitud: Longitud mínima de 1. La longitud máxima es de 255 caracteres.

Patrón: [a-zA-Z0-9\_.\-\-]+

Obligatorio: Sí

### Sintaxis de la respuesta

```
{
 "StatusCode": number
}
```

### Elementos de respuesta

Si la acción se realiza correctamente, el servicio devuelve una respuesta HTTP 200.

El servicio devuelve los datos siguientes en formato JSON.

#### [StatusCode \(p. 541\)](#)

Código de estado HTTP que indica el resultado de la operación.

Type: Entero

Rango válido: Valor mínimo de 0.

## Errores

#### [AccessDeniedException](#)

No tiene autorización para realizar la acción.

- Código de estado HTTP: 400  
**InternalServerError**  
Amazon Lex ha tenido un problema de servicio. Pruebe la llamada de nuevo.
- Código de estado HTTP: 500  
**InvalidParameterException**  
El parámetro de entrada infringió una restricción. Valide el parámetro antes de llamar a la operación de la API de nuevo.
- Código de estado HTTP: 400  
**ProvisionedThroughputExceededException**  
El número de solicitudes ha superado su límite de rendimiento. Si necesita aumentar este límite, póngase en contacto con Amazon Rekognition.
- Código de estado HTTP: 400  
**ResourceNotFoundException**  
El recurso especificado en la solicitud no se encuentra.
- Código de estado HTTP: 400  
**ThrottlingException**  
Amazon Lex no puede procesar temporalmente la solicitud. Pruebe la llamada de nuevo.
- Código de estado HTTP: 500

## Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- [AWS Command Line Interface](#)
- [SDK de AWS para .NET](#)
- [AWS SDK para C++](#)
- [AWS SDK para Go](#)
- [AWSSDK para Java V2](#)
- [AWS SDK para JavaScript](#)
- [SDK de AWS para PHP V3](#)
- [SDK de AWS para Python](#)
- [SDK de AWS para Ruby V3](#)

## DeleteDataset

Elimina un conjunto de datos existente de etiquetas personalizadas de Amazon Rekognition. Eliminar un conjunto de datos puede llevar tiempo. Usar [DescribeDataset \(p. 559\)](#) para comprobar el estado actual. El conjunto de datos se sigue eliminando si el valor de `statuses` es `DELETE_IN_PROGRESS`. Si intenta acceder al conjunto de datos después de eliminarlo, obtendrá una `ResourceNotFoundException`.

No se puede eliminar un conjunto de datos mientras se está creando (`status=CREATE_IN_PROGRESS`) o si el conjunto de datos se está actualizando (`Status=UPDATE_IN_PROGRESS`).

Esta operación requiere permisos para realizar la acción `rekognition:DeleteDataset`.

### Sintaxis de la solicitud

```
{
 "DatasetArn": "string"
}
```

### Parámetros de solicitud

La solicitud acepta los siguientes datos en formato JSON.

#### DatasetArn (p. 543)

El ARN del conjunto de datos de etiquetas personalizadas de Amazon Rekognition que desea eliminar.

Type: Cadena

Restricciones de longitud: Longitud mínima de 20. La longitud máxima es de 2048 caracteres.

Patrón: (^arn:[a-z\d-]+:rekognition:[a-z\d-]+:\d{12}:project\/[a-zA-Z0-9\_.\-\\_]{1,255}\/dataset\/(train|test)\/[0-9]+\\$)

Obligatorio: Sí

### Elementos de respuesta

Si la acción se realiza correctamente, el servicio devuelve una respuesta HTTP 200 con un cuerpo HTTP vacío.

### Errores

#### AccessDeniedException

No tiene autorización para realizar la acción.

Código de estado HTTP: 400

#### InternalServerError

Amazon Lex ha tenido un problema de servicio. Pruebe la llamada de nuevo.

Código de estado HTTP: 500

#### InvalidParameterException

El parámetro de entrada infringió una restricción. Valide el parámetro antes de llamar a la operación de la API de nuevo.

Código de estado HTTP: 400

LimitExceededException

Se ha superado un límite de servicio de Amazon Rekognition. Por ejemplo, si inicia demasiados trabajos de Amazon Rekognition Video simultáneamente, llama para iniciar operaciones (`StartLabelDetection`, por ejemplo) elevará una `LimitExceededException` excepción (código de estado HTTP: 400) hasta que el número de trabajos ejecutados simultáneamente se encuentre por debajo del límite de servicio de Amazon Rekognition.

Código de estado HTTP: 400

ProvisionedThroughputExceededException

El número de solicitudes ha superado su límite de rendimiento. Si necesita aumentar este límite, póngase en contacto con Amazon Rekognition.

Código de estado HTTP: 400

ResourceInUseException

El recurso especificado ya se está utilizando.

Código de estado HTTP: 400

ResourceNotFoundException

No se encuentra el recurso especificado en la solicitud.

Código de estado HTTP: 400

ThrottlingException

Amazon Lex no puede procesar temporalmente la solicitud. Pruebe la llamada de nuevo.

Código de estado HTTP: 500

## Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- [AWS Command Line Interface](#)
- [SDK de AWS para .NET](#)
- [AWS SDK para C++](#)
- [AWS SDK para Go](#)
- [AWSSDK para Java V2](#)
- [AWS SDK para JavaScript](#)
- [SDK de AWS para PHP V3](#)
- [SDK de AWS para Python](#)
- [SDK de AWS para Ruby V3](#)

## DeleteFaces

Elimina rostros de una colección. Especifica un ID de colección y una matriz de identificadores de caras que se van a quitar de la colección.

Esta operación requiere permisos para realizar la acción `rekognition:DeleteFaces`.

### Sintaxis de la solicitud

```
{
 "CollectionId": "string",
 "FaceIds": ["string"]
}
```

### Parámetros de solicitud

La solicitud acepta los siguientes datos en formato JSON.

#### [CollectionId \(p. 545\)](#)

Colección de la que se eliminan las caras específicas.

Type: Cadena

Restricciones de longitud: Longitud mínima de 1. La longitud máxima es de 255 caracteres.

Patrón: [ a-zA-Z0-9\_.\-\+ ]+

Obligatorio: Sí

#### [FacetIds \(p. 545\)](#)

Matriz de identificadores de caras que se van a eliminar.

Type: Matriz de cadenas

Miembros de la matriz: Número mínimo de 1 elemento. Número máximo de 4096 elementos.

Patrón: [ 0-9a-f ]{8}-[ 0-9a-f ]{4}-[ 0-9a-f ]{4}-[ 0-9a-f ]{4}-[ 0-9a-f ]{12}

Obligatorio: Sí

### Sintaxis de la respuesta

```
{
 "DeletedFaces": ["string"]
}
```

### Elementos de respuesta

Si la acción se realiza correctamente, el servicio devuelve una respuesta HTTP 200.

El servicio devuelve los datos siguientes en formato JSON.

#### [DeletedFaces \(p. 545\)](#)

Una matriz de cadenas (ID de los rostros que se eliminaron).

Type: Matriz de cadenas

Miembros de la matriz: Número mínimo de 1 elemento. Número máximo de 4096 elementos.

Patrón: [0-9a-f]{8}-[0-9a-f]{4}-[0-9a-f]{4}-[0-9a-f]{4}-[0-9a-f]{12}

## Errores

AccessDeniedException

No tiene autorización para realizar la acción.

Código de estado HTTP: 400

InternalServerError

Amazon Lex ha tenido un problema de servicio. Pruebe la llamada de nuevo.

Código de estado HTTP: 500

InvalidParameterException

El parámetro de entrada infringió una restricción. Valide el parámetro antes de llamar a la operación de la API de nuevo.

Código de estado HTTP: 400

ProvisionedThroughputExceededException

El número de solicitudes ha superado su límite de rendimiento. Si necesita aumentar este límite, póngase en contacto con Amazon Rekognition.

Código de estado HTTP: 400

ResourceNotFoundException

No se encuentra el recurso especificado en la solicitud.

Código de estado HTTP: 400

ThrottlingException

Amazon Lex no puede procesar temporalmente la solicitud. Pruebe la llamada de nuevo.

Código de estado HTTP: 500

## Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- [AWS Command Line Interface](#)
- [SDK de AWS para .NET](#)
- [AWS SDK para C++](#)
- [AWS SDK para Go](#)
- [AWSSDK para Java V2](#)
- [AWS SDK para JavaScript](#)
- [SDK de AWS para PHP V3](#)
- [SDK de AWS para Python](#)

- [SDK de AWS para Ruby V3](#)

## DeleteProject

Elimina un proyecto de etiquetas personalizadas de Amazon Rekognition. Para eliminar un proyecto, primero debe eliminar todos los modelos asociados al proyecto. Para eliminar un modelo, consulte[DeleteProjectVersion \(p. 551\)](#).

`DeleteProject`Es una operación asíncrona. Para comprobar si se ha eliminado el proyecto, llame[DescribeProjects \(p. 561\)](#). El proyecto se elimina cuando el proyecto ya no aparece en la respuesta.

Esta operación requiere permisos para realizar la acción `rekognition:DeleteProject`.

### Sintaxis de la solicitud

```
{
 "ProjectArn": "string"
}
```

### Parámetros de solicitud

La solicitud acepta los siguientes datos en formato JSON.

#### [ProjectArn \(p. 548\)](#)

El nombre de recurso de Amazon (ARN) del proyecto que desea eliminar.

Type: Cadena

Restricciones de longitud: Longitud mínima de 20. La longitud máxima es de 2048 caracteres.

Patrón: (^arn:[a-zA-Z\d-]+:rekognition:[a-zA-Z\d-]+\d{12}:project\/[a-zA-Z0-9\_.\-\\_]{1,255}\/[0-9]+\$)

Obligatorio: Sí

### Sintaxis de la respuesta

```
{
 "Status": "string"
}
```

### Elementos de respuesta

Si la acción se realiza correctamente, el servicio devuelve una respuesta HTTP 200.

El servicio devuelve los datos siguientes en formato JSON.

#### [Status \(p. 548\)](#)

El estado actual de la operación de eliminación de proyecto.

Type: Cadena

Valores válidos: CREATING | CREATED | DELETING

## Errores

AccessDeniedException

No tiene autorización para realizar la acción.

Código de estado HTTP: 400

InternalServerError

Amazon Lex ha tenido un problema de servicio. Pruebe la llamada de nuevo.

Código de estado HTTP: 500

InvalidParameterException

El parámetro de entrada infringió una restricción. Valide el parámetro antes de llamar a la operación de la API de nuevo.

Código de estado HTTP: 400

ProvisionedThroughputExceededException

El número de solicitudes ha superado su límite de rendimiento. Si necesita aumentar este límite, póngase en contacto con Amazon Rekognition.

Código de estado HTTP: 400

ResourceInUseException

El recurso especificado ya se está utilizando.

Código de estado HTTP: 400

ResourceNotFoundException

El recurso especificado en la solicitud no se encuentra.

Código de estado HTTP: 400

ThrottlingException

Amazon Lex no puede procesar temporalmente la solicitud. Pruebe la llamada de nuevo.

Código de estado HTTP: 500

## Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- [AWS Command Line Interface](#)
- [SDK de AWS para .NET](#)
- [AWS SDK para C++](#)
- [AWS SDK para Go](#)
- [AWSSDK para Java V2](#)
- [AWS SDK para JavaScript](#)
- [SDK de AWS para PHP V3](#)
- [SDK de AWS para Python](#)
- [SDK de AWS para Ruby V3](#)



## DeleteProjectVersion

Elimina un modelo de etiquetas personalizadas de Amazon Rekognition.

No puede eliminar un modelo si está en ejecución o si está en formación. Para comprobar el estado de un modelo, use el campo devuelto desde [DescribeProjectVersions \(p. 564\)](#). Para detener una llamada a un modelo en ejecución [StopProjectVersion \(p. 722\)](#). Si el modelo está entrenando, espere hasta que termine.

Esta operación requiere permisos para realizar la acción `rekognition:DeleteProjectVersion`.

### Sintaxis de la solicitud

```
{
 "ProjectVersionArn": "string"
}
```

### Parámetros de solicitud

La solicitud acepta los siguientes datos en formato JSON.

#### [ProjectVersionArn \(p. 551\)](#)

El nombre de recurso de Amazon (ARN) de la versión del modelo que desea eliminar.

Type: Cadena

Restricciones de longitud: Longitud mínima de 20. La longitud máxima es de 2048 caracteres.

Patrón: (^arn:[a-zA-Z\d-]+:rekognition:[a-zA-Z\d-]+:\d{12}:project\//[a-zA-Z0-9\_.\-\\_]{1,255}\version\//[a-zA-Z0-9\_.\-\\_]{1,255}\/[0-9]+\$)

Obligatorio: Sí

### Sintaxis de la respuesta

```
{
 "Status": "string"
}
```

### Elementos de respuesta

Si la acción se realiza correctamente, el servicio devuelve una respuesta HTTP 200.

El servicio devuelve los datos siguientes en formato JSON.

#### [Status \(p. 551\)](#)

El estado de la operación de eliminación.

Type: Cadena

Valores válidos: TRAINING\_IN\_PROGRESS | TRAINING\_COMPLETED | TRAINING\_FAILED | STARTING | RUNNING | FAILED | STOPPING | STOPPED | DELETING

## Errores

AccessDeniedException

No tiene autorización para realizar la acción.

Código de estado HTTP: 400

InternalServerError

Amazon Lex ha tenido un problema de servicio. Pruebe la llamada de nuevo.

Código de estado HTTP: 500

InvalidParameterException

El parámetro de entrada infringió una restricción. Valide el parámetro antes de llamar a la operación de la API de nuevo.

Código de estado HTTP: 400

ProvisionedThroughputExceededException

El número de solicitudes ha superado su límite de rendimiento. Si necesita aumentar este límite, póngase en contacto con Amazon Rekognition.

Código de estado HTTP: 400

ResourceInUseException

El recurso especificado ya se está utilizando.

Código de estado HTTP: 400

ResourceNotFoundException

El recurso especificado en la solicitud no se encuentra.

Código de estado HTTP: 400

ThrottlingException

Amazon Lex no puede procesar temporalmente la solicitud. Pruebe la llamada de nuevo.

Código de estado HTTP: 500

## Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- [AWS Command Line Interface](#)
- [SDK de AWS para .NET](#)
- [AWS SDK para C++](#)
- [AWS SDK para Go](#)
- [AWSSDK para Java V2](#)
- [AWS SDK para JavaScript](#)
- [SDK de AWS para PHP V3](#)
- [SDK de AWS para Python](#)
- [SDK de AWS para Ruby V3](#)



## DeleteStreamProcessor

Elimina el procesador de streaming identificado porName. Asigna el valor paraNamecuando crea el procesador de streaming con[CreateStreamProcessor \(p. 537\)](#). Es posible que no puedas usar el mismo nombre para un procesador de streaming durante unos segundos después de llamar[DeleteStreamProcessor](#).

### Sintaxis de la solicitud

```
{
 "Name": "string"
}
```

### Parámetros de solicitud

La solicitud acepta los siguientes datos en formato JSON.

#### Name (p. 554)

El nombre del procesador de streaming que desea borrar.

Type: Cadena

Restricciones de longitud: Longitud mínima de 1. La longitud máxima es de 128 caracteres.

Patrón: [a-zA-Z0-9\_.\-\-]+

Obligatorio: Sí

### Elementos de respuesta

Si la acción se realiza correctamente, el servicio devuelve una respuesta HTTP 200 con un cuerpo HTTP vacío.

### Errores

#### AccessDeniedException

No tiene autorización para realizar la acción.

Código de estado HTTP: 400

#### InternalServerError

Amazon Lex ha tenido un problema de servicio. Pruebe la llamada de nuevo.

Código de estado HTTP: 500

#### InvalidArgumentException

El parámetro de entrada infringió una restricción. Valide el parámetro antes de llamar a la operación de la API de nuevo.

Código de estado HTTP: 400

#### ProvisionedThroughputExceededException

El número de solicitudes ha superado su límite de rendimiento. Si necesita aumentar este límite, póngase en contacto con Amazon Rekognition.

Código de estado HTTP: 400

ResourceInUseException

El recurso especificado ya se está utilizando.

Código de estado HTTP: 400

ResourceNotFoundException

No se encuentra el recurso especificado en la solicitud.

Código de estado HTTP: 400

ThrottlingException

Amazon Lex no puede procesar temporalmente la solicitud. Pruebe la llamada de nuevo.

Código de estado HTTP: 500

## Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- [AWS Command Line Interface](#)
- [SDK de AWS para .NET](#)
- [AWS SDK para C++](#)
- [AWS SDK para Go](#)
- [AWSSDK para Java V2](#)
- [AWS SDK para JavaScript](#)
- [SDK de AWS para PHP V3](#)
- [SDK de AWS para Python](#)
- [SDK de AWS para Ruby V3](#)

## DescribeCollection

Describe la colección especificada. Puede usar `DescribeCollection` para obtener información, como el número de caras indexadas en una colección y la versión del modelo utilizado por la colección para la detección de rostros.

Para obtener más información, consulte [Descripción de una colección \(p. 196\)](#).

### Sintaxis de la solicitud

```
{
 "CollectionId": "string"
}
```

### Parámetros de solicitud

La solicitud acepta los siguientes datos en formato JSON.

#### [CollectionId \(p. 556\)](#)

El ID de la colección que describir.

Type: Cadena

Limitaciones de longitud: Longitud mínima de 1. La longitud máxima es de 255 caracteres.

Patrón: [ a-zA-Z0-9\_.\-\-]+

Obligatorio: Sí

### Sintaxis de la respuesta

```
{
 "CollectionARN": "string",
 "CreationTimestamp": number,
 "FaceCount": number,
 "FaceModelVersion": "string"
}
```

### Elementos de respuesta

Si la acción se realiza correctamente, el servicio devuelve una respuesta HTTP 200.

El servicio devuelve los datos siguientes en formato JSON.

#### [CollectionARN \(p. 556\)](#)

El nombre de recurso de Amazon (ARN) de la colección.

Type: Cadena

#### [CreationTimestamp \(p. 556\)](#)

El número de milisegundos desde el tamaño 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.

Type: Marca temporal

### [FaceCount \(p. 556\)](#)

El número de rostros que se indexan en la colección. Para indexar caras en una colección, utilice [IndexFaces \(p. 644\)](#).

Type: Long

Rango válido: Valor mínimo de 0.

### [FaceModelVersion \(p. 556\)](#)

La versión del modelo de rostro que utiliza la colección para la detección de rostros.

Para obtener más información, consulte [Control de versiones del modelo \(p. 10\)](#).

Type: Cadena

## Errores

### AccessDeniedException

No tiene autorización para realizar la acción.

Código de estado HTTP: 400

### InternalServerError

Amazon Lex ha tenido un problema de servicio. Pruebe la llamada de nuevo.

Código de estado HTTP: 500

### InvalidParameterException

El parámetro de entrada infringió una restricción. Valide el parámetro antes de llamar a la operación de la API de nuevo.

Código de estado HTTP: 400

### ProvisionedThroughputExceededException

El número de solicitudes ha superado su límite de rendimiento. Si necesita aumentar este límite, póngase en contacto con Amazon Rekognition.

Código de estado HTTP: 400

### ResourceNotFoundException

No se encuentra el recurso especificado en la solicitud.

Código de estado HTTP: 400

### ThrottlingException

Amazon Lex no puede procesar temporalmente la solicitud. Pruebe la llamada de nuevo.

Código de estado HTTP: 500

## Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- [AWS Command Line Interface](#)

- [SDK de AWS para .NET](#)
- [AWS SDK para C++](#)
- [AWS SDK para Go](#)
- [AWSSDK para Java V2](#)
- [AWS SDK para JavaScript](#)
- [SDK de AWS para PHP V3](#)
- [SDK de AWS para Python](#)
- [SDK de AWS para Ruby V3](#)

## DescribeDataset

Describe un conjunto de datos de etiquetas personalizadas de Amazon Rekognition. Puede obtener información como el estado actual de un conjunto de datos y estadísticas sobre las imágenes y etiquetas de un conjunto de datos.

Esta operación requiere permisos para realizar la acción `rekognition:DescribeDataset`.

### Sintaxis de la solicitud

```
{
 "DatasetArn": "string"
}
```

### Parámetros de solicitud

La solicitud acepta los siguientes datos en formato JSON.

#### [DatasetArn \(p. 559\)](#)

El nombre de recurso de Amazon (ARN) del conjunto de datos que desea describir.

Type: Cadena

Restricciones de longitud: Longitud mínima de 20. La longitud máxima es de 2048 caracteres.

Patrón: (^arn:[a-zA-Z\d-]+:rekognition:[a-zA-Z\d-]+\d{12}:project\/[a-zA-Z0-9\_.\-\\_]{1,255}\dataset\/(train|test)\/[0-9]+\$)

Obligatorio: Sí

### Sintaxis de la respuesta

```
{
 "DatasetDescription": {
 "CreationTimestamp": number,
 "DatasetStats": {
 "ErrorEntries": number,
 "LabeledEntries": number,
 "TotalEntries": number,
 "TotalLabels": number
 },
 "LastUpdatedTimestamp": number,
 "Status": "string",
 "StatusMessage": "string",
 "StatusMessageCode": "string"
 }
}
```

### Elementos de respuesta

Si la acción se realiza correctamente, el servicio devuelve una respuesta HTTP 200.

El servicio devuelve los datos siguientes en formato JSON.

#### [DatasetDescription \(p. 559\)](#)

Descripción del conjunto de datos.

Tipo: objeto [DatasetDescription](#) (p. 755)

## Errores

AccessDeniedException

No tiene autorización para realizar la acción.

Código de estado HTTP: 400

InternalServerError

Amazon Lex ha tenido un problema de servicio. Pruebe la llamada de nuevo.

Código de estado HTTP: 500

InvalidParameterException

El parámetro de entrada infringió una restricción. Valide su parámetro antes de llamar a la operación de la API de nuevo.

Código de estado HTTP: 400

ProvisionedThroughputExceededException

El número de solicitudes ha superado su límite de rendimiento. Si necesita aumentar este límite, póngase en contacto con Amazon Rekognition.

Código de estado HTTP: 400

ResourceNotFoundException

No se encuentra el recurso especificado en la solicitud.

Código de estado HTTP: 400

ThrottlingException

Amazon Lex no puede procesar temporalmente la solicitud. Pruebe la llamada de nuevo.

Código de estado HTTP: 500

## Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- [AWS Command Line Interface](#)
- [SDK de AWS para .NET](#)
- [AWS SDK para C++](#)
- [AWS SDK para Go](#)
- [AWSSDK para Java V2](#)
- [AWS SDK para JavaScript](#)
- [SDK de AWS para PHP V3](#)
- [SDK de AWS para Python](#)
- [SDK de AWS para Ruby V3](#)

## DescribeProjects

Obtiene información sobre los proyectos de etiquetas personalizadas de Amazon Rekognition.

Esta operación requiere permisos para realizar la acción `rekognition:DescribeProjects`.

### Sintaxis de la solicitud

```
{
 "MaxResults": number,
 "NextToken": "string",
 "ProjectNames": ["string"]
}
```

### Parámetros de solicitud

La solicitud acepta los siguientes datos en formato JSON.

#### [MaxResults \(p. 561\)](#)

El número máximo de resultados que devolver por llamada paginada. El valor más grande que puede especificar es 100. Si especifica un valor mayor que 100, se produce un error de `ValidationException`. El valor predeterminado es 100.

Type: Entero

Rango válido: Valor mínimo de 1. Valor máximo de 100.

Obligatorio: No

#### [NextToken \(p. 561\)](#)

Si la respuesta anterior estaba incompleta (porque hay más resultados que recuperar), Amazon Rekognition Custom Labels devuelve un token de paginación en la respuesta. Puede utilizar este token de paginación para recuperar el siguiente conjunto de resultados.

Type: Cadena

Restricciones de longitud: La longitud máxima es de 1024 caracteres.

Obligatorio: No

#### [ProjectNames \(p. 561\)](#)

Lista de los proyectos que desea que describa las etiquetas personalizadas de Amazon Rekognition. Si no especifica un valor, la respuesta incluye descripciones de todos los proyectos de su cuenta de AWS.

Type: Matriz de cadenas

Miembros de la matriz: Número mínimo de 1 elemento. Número máximo de 10 elementos.

Restricciones de longitud: Longitud mínima de 1. La longitud máxima es de 255 caracteres.

Patrón: [a-zA-Z0-9\_.\-\+]+

Obligatorio: No

## Sintaxis de la respuesta

```
{
 "NextToken": "string",
 "ProjectDescriptions": [
 {
 "CreationTimestamp": number,
 "Datasets": [
 {
 "CreationTimestamp": number,
 "DatasetArn": "string",
 "DatasetType": "string",
 "Status": "string",
 "StatusMessage": "string",
 "StatusMessageCode": "string"
 }
],
 "ProjectArn": "string",
 "Status": "string"
 }
]
}
```

## Elementos de respuesta

Si la acción se realiza correctamente, el servicio devuelve una respuesta HTTP 200.

El servicio devuelve los datos siguientes en formato JSON.

### [NextToken](#) (p. 562)

Si la respuesta anterior estaba incompleta (porque hay más resultados que recuperar), Amazon Rekognition Custom Labels devuelve un token de paginación en la respuesta. Puede utilizar este token de paginación para recuperar el siguiente conjunto de resultados.

Type: Cadena

Restricciones de longitud: La longitud máxima es de 1024 caracteres.

### [ProjectDescriptions](#) (p. 562)

Lista de descripciones de proyectos. La lista está ordenada por la fecha y hora en que se crean los proyectos.

Type: Matriz de[ProjectDescription](#) (p. 807)objects

## Errores

### AccessDeniedException

No tiene autorización para realizar la acción.

Código de estado HTTP: 400

### InternalServerError

Amazon Lex ha tenido un problema de servicio. Pruebe la llamada de nuevo.

Código de estado HTTP: 500

**InvalidPaginationTokenException**

El token de paginación de la solicitud no es válido.

Código de estado HTTP: 400

**InvalidParameterException**

El parámetro de entrada infringió una restricción. Valide el parámetro antes de llamar a la operación de la API de nuevo.

Código de estado HTTP: 400

**ProvisionedThroughputExceededException**

El número de solicitudes ha superado su límite de rendimiento. Si necesita aumentar este límite, póngase en contacto con Amazon Rekognition.

Código de estado HTTP: 400

**ThrottlingException**

Amazon Lex no puede procesar temporalmente la solicitud. Pruebe la llamada de nuevo.

Código de estado HTTP: 500

## Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- [AWS Command Line Interface](#)
- [SDK de AWS para .NET](#)
- [AWS SDK para C++](#)
- [AWS SDK para Go](#)
- [AWSSDK para Java V2](#)
- [AWS SDK para JavaScript](#)
- [SDK de AWS para PHP V3](#)
- [SDK de AWS para Python](#)
- [SDK de AWS para Ruby V3](#)

## DescribeProjectVersions

Enumera y describe las versiones de un modelo en un proyecto de etiquetas personalizadas de Amazon Rekognition. Puede especificar hasta 10 versiones de modelo en `ProjectVersionArns`. Si no especifica un valor, se devuelve las descripciones de todas las versiones de modelo del proyecto.

Esta operación requiere permisos para realizar la acción `rekognition:DescribeProjectVersions`.

### Sintaxis de la solicitud

```
{
 "MaxResults": number,
 "NextToken": "string",
 "ProjectArn": "string",
 "VersionNames": ["string"]
}
```

### Parámetros de solicitud

La solicitud acepta los siguientes datos en formato JSON.

#### [MaxResults \(p. 564\)](#)

El número máximo de resultados que devolver por llamada paginada. El valor más grande que puede especificar es 100. Si especifica un valor mayor que 100, se produce un error de `ValidationException`. El valor predeterminado es 100.

Type: Entero

Rango válido: Valor mínimo de 1. Valor máximo de 100.

Obligatorio: No

#### [NextToken \(p. 564\)](#)

Si la respuesta anterior estaba incompleta (porque hay más resultados que recuperar), Amazon Rekognition Custom Labels devuelve un token de paginación en la respuesta. Puede utilizar este token de paginación para recuperar el siguiente grupo de resultados.

Type: Cadena

Restricciones de longitud: La longitud máxima es de 1024 caracteres.

Obligatorio: No

#### [ProjectArn \(p. 564\)](#)

El nombre de recurso de Amazon (ARN) del proyecto que contiene los modelos que desea describir.

Type: Cadena

Restricciones de longitud: Longitud mínima de 20. La longitud máxima es de 2048 caracteres.

Patrón: (`^arn:[a-z\d-]+:rekognition:[a-z\d-]+\:\d{12}:project\/[a-zA-Z0-9_.\-\_]{1,255}\/[0-9]+\$`)

Obligatorio: Sí

#### [VersionNames \(p. 564\)](#)

Lista de nombres de versiones de modelo que desea describir. Puede añadir hasta 10 nombres de versiones de modelo a la lista. Si no especifica un valor, se devuelve todas las descripciones

del modelo. El nombre de una versión forma parte de un ARN de modelo (ProjectVersion). Por ejemplo, `my-model.2020-01-21T09.10.15` es el nombre de la versión del siguiente ARN.`arn:aws:rekognition:us-east-1:123456789012:project/getting-started/version/my-model.2020-01-21T09.10.15/1234567890123`.

Type: Matriz de cadenas

Miembros de la matriz: Número mínimo de 1 elemento. Número máximo de 10 elementos.

Restricciones de longitud: Longitud mínima de 1. La longitud máxima es de 255 caracteres.

Patrón: `[a-zA-Z0-9_.\-\+]`

Obligatorio: No

## Sintaxis de la respuesta

```
{
 "NextToken": "string",
 "ProjectVersionDescriptions": [
 {
 "BillableTrainingTimeInSeconds": number,
 "CreationTimestamp": number,
 "EvaluationResult": {
 "F1Score": number,
 "Summary": {
 "S3Object": {
 "Bucket": "string",
 "Name": "string",
 "Version": "string"
 }
 }
 },
 "KmsKeyId": "string",
 "ManifestSummary": {
 "S3Object": {
 "Bucket": "string",
 "Name": "string",
 "Version": "string"
 }
 },
 "MinInferenceUnits": number,
 "OutputConfig": {
 "S3Bucket": "string",
 "S3KeyPrefix": "string"
 },
 "ProjectVersionArn": "string",
 "Status": "string",
 "StatusMessage": "string",
 "TestingDataResult": {
 "Input": {
 "Assets": [
 {
 "GroundTruthManifest": {
 "S3Object": {
 "Bucket": "string",
 "Name": "string",
 "Version": "string"
 }
 }
 }
]
 },
 "AutoCreate": boolean
 }
 }
]
}
```

```
 },
 "Output": {
 "Assets": [
 {
 "GroundTruthManifest": {
 "S3Object": {
 "Bucket": "string",
 "Name": "string",
 "Version": "string"
 }
 }
 }
],
 "AutoCreate": boolean
 },
 "Validation": {
 "Assets": [
 {
 "GroundTruthManifest": {
 "S3Object": {
 "Bucket": "string",
 "Name": "string",
 "Version": "string"
 }
 }
 }
]
 }
 },
 "TrainingDataResult": {
 "Input": {
 "Assets": [
 {
 "GroundTruthManifest": {
 "S3Object": {
 "Bucket": "string",
 "Name": "string",
 "Version": "string"
 }
 }
 }
]
 },
 "Output": {
 "Assets": [
 {
 "GroundTruthManifest": {
 "S3Object": {
 "Bucket": "string",
 "Name": "string",
 "Version": "string"
 }
 }
 }
]
 }
 },
 "Validation": {
 "Assets": [
 {
 "GroundTruthManifest": {
 "S3Object": {
 "Bucket": "string",
 "Name": "string",
 "Version": "string"
 }
 }
 }
]
 }
},
```

```
 }
]
},
"TrainingEndTimestamp": number
]
}
```

## Elementos de respuesta

Si la acción se realiza correctamente, el servicio devuelve una respuesta HTTP 200.

El servicio devuelve los datos siguientes en formato JSON.

### [NextToken](#) (p. 565)

Si la respuesta anterior estaba incompleta (porque hay más resultados que recuperar), Amazon Rekognition Custom Labels devuelve un token de paginación en la respuesta. Puede utilizar este token de paginación para recuperar el siguiente grupo de resultados.

Type: Cadena

Restricciones de longitud: La longitud máxima es de 1024 caracteres.

### [ProjectVersionDescriptions](#) (p. 565)

Lista de descripciones de modelos. La lista se ordena por la fecha y hora de creación de las versiones del modelo, de última a temprana.

Type: Matriz de[ProjectVersionDescription](#) (p. 808)objects

## Errores

### AccessDeniedException

No tiene autorización para realizar la acción.

Código de estado HTTP: 400

### InternalServerError

Amazon Lex ha tenido un problema de servicio. Pruebe la llamada de nuevo.

Código de estado HTTP: 500

### InvalidPaginationTokenException

El token de paginación de la solicitud no es válido.

Código de estado HTTP: 400

### InvalidParameterException

El parámetro de entrada infringió una restricción. Valide el parámetro antes de llamar a la operación de la API de nuevo.

Código de estado HTTP: 400

### ProvisionedThroughputExceededException

El número de solicitudes ha superado su límite de rendimiento. Si necesita aumentar este límite, póngase en contacto con Amazon Rekognition.

- Código de estado HTTP: 400  
`ResourceNotFoundException`
  - No se encuentra el recurso especificado en la solicitud.
- Código de estado HTTP: 400  
`ThrottlingException`
  - Amazon Lex no puede procesar temporalmente la solicitud. Pruebe la llamada de nuevo.
- Código de estado HTTP: 500

## Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- [AWS Command Line Interface](#)
- [SDK de AWS para .NET](#)
- [AWS SDK para C++](#)
- [AWS SDK para Go](#)
- [AWSSDK para Java V2](#)
- [AWS SDK para JavaScript](#)
- [SDK de AWS para PHP V3](#)
- [SDK de AWS para Python](#)
- [SDK de AWS para Ruby V3](#)

## DescribeStreamProcessor

Proporciona información acerca de un procesador de streaming creado por [CreateStreamProcessor \(p. 537\)](#). Puede obtener información sobre los flujos de entrada y salida, los parámetros de entrada para el reconocimiento facial que se está realizando y el estado actual del procesador de secuencias.

### Sintaxis de la solicitud

```
{
 "Name": "string"
}
```

### Parámetros de solicitud

La solicitud acepta los siguientes datos en formato JSON.

#### Name (p. 569)

Nombre del procesador de secuencias para el que desea información.

Type: Cadena

Restricciones de longitud: Longitud mínima de 1. La longitud máxima es de 128 caracteres.

Patrón: [a-zA-Z0-9\_.\-\-]+

Obligatorio: Sí

### Sintaxis de la respuesta

```
{
 "CreationTimestamp": number,
 "Input": {
 "KinesisVideoStream": {
 "Arn": "string"
 }
 },
 "LastUpdateTimestamp": number,
 "Name": "string",
 "Output": {
 "KinesisDataStream": {
 "Arn": "string"
 }
 },
 "RoleArn": "string",
 "Settings": {
 "FaceSearch": {
 "CollectionId": "string",
 "FaceMatchThreshold": number
 }
 },
 "Status": "string",
 "StatusMessage": "string",
 "StreamProcessorArn": "string"
}
```

## Elementos de respuesta

Si la acción se realiza correctamente, el servicio devuelve una respuesta HTTP 200.

El servicio devuelve los datos siguientes en formato JSON.

### [CreationTimestamp \(p. 569\)](#)

Fecha y hora en que se creó el procesador de secuencias

Type: Marca temporal

### [Input \(p. 569\)](#)

Transmisión de vídeo de Kinesis que proporciona la transmisión de vídeo de origen.

Tipo: objeto [StreamProcessorInput \(p. 829\)](#)

### [LastUpdateTimestamp \(p. 569\)](#)

La hora, en formato Unix, se actualizó por última vez el procesador de secuencias. Por ejemplo, cuando el procesador de secuencias pasa de un estado en ejecución a un estado de error o cuando el usuario inicia o detiene el procesador de secuencias.

Type: Marca temporal

### [Name \(p. 569\)](#)

Nombre del procesador de streaming.

Type: Cadena

Restricciones de longitud: Longitud mínima de 1. La longitud máxima es de 128 caracteres.

Patrón: [a-zA-Z0-9\_.\-\\_]+

### [Output \(p. 569\)](#)

Flujo de datos de Kinesis en el que Amazon Rekognition Video pone los resultados del análisis.

Tipo: objeto [StreamProcessorOutput \(p. 830\)](#)

### [RoleArn \(p. 569\)](#)

El ARN del rol de IAM que permite el acceso al procesador de streaming.

Type: Cadena

Patrón: arn:aws:iam::\d{12}:role/[a-zA-Z0-9+=,.@\-\\_/\_]+

### [Settings \(p. 569\)](#)

Parámetros de entrada de reconocimiento facial que utiliza el procesador de secuencias. Incluye la colección que se utilizará para el reconocimiento facial y los atributos de cara que se van a detectar.

Tipo: objeto [StreamProcessorSettings \(p. 831\)](#)

### [Status \(p. 569\)](#)

El estado actual del procesador de streaming.

Type: Cadena

Valores válidos: STOPPED | STARTING | RUNNING | FAILED | STOPPING

### [StatusMessage \(p. 569\)](#)

El mensaje de estado detallado acerca del procesador de streaming.

Type: Cadena

[StreamProcessorArn \(p. 569\)](#)

El ARN del procesador de streaming.

Type: Cadena

Patrón: (^arn:[a-z\d-]+:rekognition:[a-z\d-]+:\d{12}:streamprocessor\/.+)\$)

## Errores

AccessDeniedException

No tiene autorización para realizar la acción.

Código de estado HTTP: 400

InternalServerError

Amazon Lex ha tenido un problema de servicio. Pruebe la llamada de nuevo.

Código de estado HTTP: 500

InvalidArgumentException

El parámetro de entrada infringió una restricción. Valide el parámetro antes de llamar a la operación de la API de nuevo.

Código de estado HTTP: 400

ProvisionedThroughputExceededException

El número de solicitudes ha superado su límite de rendimiento. Si necesita aumentar este límite, póngase en contacto con Amazon Rekognition.

Código de estado HTTP: 400

ResourceNotFoundException

El recurso especificado en la solicitud no se encuentra.

Código de estado HTTP: 400

ThrottlingException

Amazon Lex no puede procesar temporalmente la solicitud. Pruebe la llamada de nuevo.

Código de estado HTTP: 500

## Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- [AWS Command Line Interface](#)
- [SDK de AWS para .NET](#)
- [AWS SDK para C++](#)
- [AWS SDK para Go](#)
- [AWSSDK para Java V2](#)
- [AWS SDK para JavaScript](#)

- [SDK de AWS para PHP V3](#)
- [SDK de AWS para Python](#)
- [SDK de AWS para Ruby V3](#)

## DetectCustomLabels

Detecta etiquetas personalizadas en una imagen suministrada mediante un modelo de etiquetas personalizadas de Amazon Rekognition.

Especifica qué versión de una versión de modelo se va a utilizar mediante la `ProjectVersionArn` parámetro de entrada.

Puede transferir la imagen de entrada como bytes de imagen codificados en base64 o como referencia a una imagen de un bucket de Amazon S3. Si utiliza la CLI de AWS para llamar a las operaciones de Amazon Rekognition, no es posible transferir bytes de imágenes. La imagen debe ser un archivo de formato PNG o JPEG.

Para cada objeto que detecta la versión del modelo en una imagen, la API devuelve un (`CustomLabel`) en una matriz (`CustomLabels`). Cada `CustomLabel` objeto proporciona el nombre de la etiqueta (`Name`), el grado de confianza de que la imagen contenga el objeto (`Confidence`) e información de ubicación de objetos, si existe, para la etiqueta de la imagen (`Geometry`).

Para filtrar las etiquetas que se devuelven, especifique un valor para `MinConfidence`. `DetectCustomLabels` solo devuelve etiquetas con una confianza superior al valor especificado. El valor de `MinConfidence` se asigna a los valores umbrales asumidos creados durante el entrenamiento. Para obtener más información, consulte [Umbral asumido](#). Las etiquetas personalizadas de Amazon Rekognition expresan un umbral supuesto como un valor de coma flotante entre 0 y 1. Rango `MinConfidence` normaliza el umbral supuesto a un valor porcentual (0-100). Respuestas de confianza de `DetectCustomLabels` se devuelven también como porcentaje. Puede usar `MinConfidence` para cambiar la precisión y la retirada de su modelo. Para obtener más información, consulte [Análisis de una imagen](#).

Si no especifica un valor para `MinConfidence`, `DetectCustomLabels` devuelve etiquetas en función del umbral supuesto de cada etiqueta.

Se trata de una operación de API sin estado. Es decir, la operación no persiste ningún dato.

Esta operación requiere permisos para realizar la acción `rekognition:DetectCustomLabels`.

Para obtener más información, consulte [Análisis de una imagen](#).

## Sintaxis de la solicitud

```
{
 "Image": {
 "Bytes": blob,
 "S3Object": {
 "Bucket": "string",
 "Name": "string",
 "Version": "string"
 }
 },
 "MaxResults": number,
 "MinConfidence": number,
 "ProjectVersionArn": "string"
}
```

## Parámetros de solicitud

La solicitud acepta los siguientes datos en formato JSON.

### [Image \(p. 573\)](#)

Proporciona la imagen de entrada en bytes o como objeto S3.

Puede transferir bytes de imágenes a una operación API de Amazon Rekognition utilizando la `Bytes` propiedad. Por ejemplo, usaría el comando `Bytes` para transferir una imagen cargada desde un sistema de archivos local. Bytes de imagen pasados mediante el `Bytes` debe estar codificado en base64. Es posible que el código no necesite codificar bytes de imagen si utiliza un SDK de AWS para llamar a las operaciones de la API de Amazon Rekognition.

Para obtener más información, consulte [Análisis de una imagen cargada desde un sistema de archivos local \(p. 39\)](#).

Puede transferir imágenes almacenadas en un bucket de S3 a una operación de API de Amazon Rekognition utilizando la `S3Object` propiedad. Las imágenes almacenadas en un bucket de S3 no tienen que estar codificadas en base64.

La región del bucket de S3 que contiene el objeto S3 debe coincidir con la región que utiliza para las operaciones de Amazon Rekognition.

Si utiliza la CLI de AWS para llamar a las operaciones de Amazon Rekognition, no es posible transferir bytes de imágenes utilizando la propiedad `Bytes`. Debe cargar primero la imagen en un bucket de Amazon S3 y, a continuación, llamar a la operación utilizando la propiedad `S3Object`.

Para que Amazon Rekognition procese un objeto de S3, el usuario debe tener permiso para acceder al objeto de S3. Para obtener más información, consulte [Políticas de Amazon Rekognition basadas en recursos \(p. 487\)](#).

**Tipo:** objeto [Image \(p. 786\)](#)

**obligatorio:** obligatorio Sí

[MaxResults \(p. 573\)](#)

El número máximo de resultados que desea que el servicio devuelva en la respuesta. El servicio devuelve el número especificado de etiquetas de mayor confianza clasificadas de mayor confianza a menor.

**Type:** Entero

Rango válido: Valor mínimo de 0.

**obligatorio:** obligatorio No

[MinConfidence \(p. 573\)](#)

Especifica el nivel de confianza mínimo de las etiquetas que se van a devolver. `DetectCustomLabels` no devuelve ninguna etiqueta con un valor de confianza inferior al valor especificado. Si especifica un valor de 0, `DetectCustomLabels` devuelve todas las etiquetas, independientemente del umbral supuesto aplicado a cada etiqueta. Si no especifica un valor para `MinConfidence`, `DetectCustomLabels` devuelve etiquetas en función del umbral supuesto de cada etiqueta.

**Type:** Float

Rango válido: Valor mínimo de 0. Valor máximo de 100.

**obligatorio:** obligatorio No

[ProjectVersionArn \(p. 573\)](#)

El ARN de la versión de modelo que desea utilizar.

**Type:** Cadena

Restricciones de longitud: Longitud mínima de 20. La longitud máxima es de 2048 caracteres.

Patrón: (^arn:[a-z\d-]+:rekognition:[a-z\d-]+\d{12}:project\/[a-zA-Z0-9\_.\-\\_]{1,255}\version\/[a-zA-Z0-9\_.\-\\_]{1,255}\/[0-9]+\$)

obligatorio: obligatorio Sí

## Sintaxis de la respuesta

```
{
 "CustomLabels": [
 {
 "Confidence": number,
 "Geometry": {
 "BoundingBox": {
 "Height": number,
 "Left": number,
 "Top": number,
 "Width": number
 },
 "Polygon": [
 {
 "X": number,
 "Y": number
 }
]
 },
 "Name": "string"
 }
]
}
```

## Elementos de respuesta

Si la acción se realiza correctamente, el servicio devuelve una respuesta HTTP 200.

El servicio devuelve los datos siguientes en formato JSON.

### CustomLabels (p. 575)

Matriz de etiquetas personalizadas detectadas en la imagen de entrada.

Type: Matriz de [CustomLabel \(p. 753\)](#) objects

## Errores

### AccessDeniedException

No tiene autorización para realizar la acción.

Código de estado HTTP: 400

### ImageTooLargeException

La imagen de entrada tamaño supera el límite permitido. Si estás llamando [DetectProtectiveEquipment \(p. 592\)](#), el tamaño o la resolución de la imagen supera el límite permitido. Para obtener más información, consulte [Directrices y cuotas de Amazon Rekognition \(p. 847\)](#).

Código de estado HTTP: 400

InternalServerError

Amazon Lex ha tenido un problema de servicio. Pruebe la llamada de nuevo.

Código de estado HTTP: 500

InvalidImageFormatException

No se admite el formato de imagen proporcionado.

Código de estado HTTP: 400

InvalidArgumentException

El parámetro de entrada infringió una restricción. Valide el parámetro antes de llamar a la operación de la API de nuevo.

Código de estado HTTP: 400

InvalidS3ObjectException

Amazon Rekognition no puede obtener acceso al objeto de S3 especificado en la solicitud.

Código de estado HTTP: 400

LimitExceededException

Se ha superado un límite de servicio de Amazon Rekognition. Por ejemplo, si inicia demasiados trabajos de Amazon Rekognition Video simultáneamente, llama para iniciar operaciones (`StartLabelDetection`, por ejemplo) elevará un `LimitExceededException` excepción (código de estado HTTP: 400) hasta que el número de trabajos ejecutados simultáneamente se encuentre por debajo del límite de servicio de Amazon Rekognition.

Código de estado HTTP: 400

ProvisionedThroughputExceededException

El número de solicitudes ha superado su límite de rendimiento. Si necesita aumentar este límite, póngase en contacto con Amazon Rekognition.

Código de estado HTTP: 400

ResourceNotFoundException

El recurso especificado en la solicitud no se encuentra.

Código de estado HTTP: 400

ResourceNotReadyException

El recurso solicitado no está listo. Por ejemplo, esta excepción se produce cuando llama `DetectCustomLabels` con una versión de modelo que no está implementada.

Código de estado HTTP: 400

ThrottlingException

Amazon Lex no puede procesar temporalmente la solicitud. Pruebe la llamada de nuevo.

Código de estado HTTP: 500

## Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- AWS Command Line Interface
- SDK de AWS para .NET
- AWS SDK para C++
- AWS SDK para Go
- AWSSDK para Java V2
- AWS SDK para JavaScript
- SDK de AWS para PHP V3
- SDK de AWS para Python
- SDK de AWS para Ruby V3

## DetectFaces

Detecta rostros en una imagen que se proporciona como entrada.

`DetectFaces` detecta los 100 rostros de mayor tamaño en la imagen. Para cada cara detectada, la operación devuelve los detalles de la cara. Estos detalles incluyen un cuadro delimitador de la cara, un valor de confianza (que el cuadro delimitador contiene una cara) y un conjunto fijo de atributos tales como puntos de referencia faciales (por ejemplo, coordenadas de ojo y boca), presencia de barba, gafas de sol, etc.

El algoritmo de detección facial es más eficaz en caras frontales. En el caso de caras no frontales u oscuras, es posible que el algoritmo no detecte las caras o detecte caras con menor confianza.

Se pasa la imagen de entrada como bytes de imagen codificados en base64 o como referencia a una imagen de un bucket de Amazon S3. Si utiliza la AWS CLI para llamar a las operaciones de Amazon Rekognition, no es posible pasar bytes de imágenes. La imagen debe ser un archivo de formato PNG o JPEG.

### Note

Se trata de una operación de API sin estado. Es decir, la operación no persiste ningún dato.

Esta operación requiere permisos para realizar la acción `rekognition:DetectFaces`.

## Sintaxis de la solicitud

```
{
 "Attributes": ["string"],
 "Image": {
 "Bytes": blob,
 "S3Object": {
 "Bucket": "string",
 "Name": "string",
 "Version": "string"
 }
 }
}
```

## Parámetros de solicitud

La solicitud acepta los siguientes datos en formato JSON.

### Attributes (p. 578)

Una serie de atributos faciales que desea que se devuelvan. Puede ser la lista predeterminada de atributos o todos los atributos. Si no especifica un valor para `Attributes` o si especifica `["DEFAULT"]`, la API devuelve el siguiente subconjunto de atributos faciales: `BoundingBox`, `Confidence`, `Pose`, `Quality`, y `Landmarks`. Si proporcionas `["ALL"]`, se devuelven todos los atributos faciales, pero la operación tarda más en completarse.

Si proporciona ambas cosas, `["ALL", "DEFAULT"]`, el servicio utiliza un operador AND lógico para determinar qué atributos devolver (en este caso, todos los atributos).

Type: Matriz de cadenas

Valores válidos: `DEFAULT` | `ALL`

Obligatorio: No

[Image \(p. 578\)](#)

La imagen de entrada como bytes codificados en base64 o un objeto S3. Si utiliza la AWS CLI para llamar a las operaciones de Amazon Rekognition, no es posible pasar bytes de imágenes codificadas en base 64.

Si utiliza un SDK de AWS para llamar a Amazon Rekognition, es posible que no tenga que codificar en base 64 bytes de imagen pasados mediante laBytes. Para obtener más información, consulte [Especificaciones de imágenes \(p. 29\)](#).

Tipo: objeto [Image \(p. 786\)](#)

Obligatorio: Sí

## Sintaxis de la respuesta

```
{
 "FaceDetails": [
 {
 "AgeRange": {
 "High": number,
 "Low": number
 },
 "Beard": {
 "Confidence": number,
 "Value": boolean
 },
 "BoundingBox": {
 "Height": number,
 "Left": number,
 "Top": number,
 "Width": number
 },
 "Confidence": number,
 "Emotions": [
 {
 "Confidence": number,
 "Type": "string"
 }
],
 "Eyeglasses": {
 "Confidence": number,
 "Value": boolean
 },
 "EyesOpen": {
 "Confidence": number,
 "Value": boolean
 },
 "Gender": {
 "Confidence": number,
 "Value": "string"
 },
 "Landmarks": [
 {
 "Type": "string",
 "X": number,
 "Y": number
 }
],
 "MouthOpen": {
 "Confidence": number,
 "Value": boolean
 }
 }
]
}
```

```
 "Confidence": number,
 "Value": boolean
 },
 "Mustache": {
 "Confidence": number,
 "Value": boolean
 },
 "Pose": {
 "Pitch": number,
 "Roll": number,
 "Yaw": number
 },
 "Quality": {
 "Brightness": number,
 "Sharpness": number
 },
 "Smile": {
 "Confidence": number,
 "Value": boolean
 },
 "Sunglasses": {
 "Confidence": number,
 "Value": boolean
 }
}
],
"OrientationCorrection": "string"
}
```

## Elementos de respuesta

Si la acción se realiza correctamente, el servicio devuelve una respuesta HTTP 200.

El servicio devuelve los datos siguientes en formato JSON.

### [FaceDetails \(p. 579\)](#)

Detalles de cada cara encontrada en la imagen.

Type: Matrices de[FaceDetail \(p. 773\)](#)objects

### [OrientationCorrection \(p. 579\)](#)

El valor deOrientationCorrectionsiempre es nulo.

Si la imagen de entrada está en formato .jpeg, puede que contenga metadatos de formato de archivo de imagen intercambiable (EXIF) que incluyen la orientación de la imagen. Amazon Rekognition utiliza esta información de orientación para realizar la corrección de imagen. Las coordenadas del cuadro delimitador se traducen para representar las ubicaciones de los objetos después de utilizar la información de orientación de los metadatos Exif para corregir la orientación de la imagen. Las imágenes en formato .png no contienen metadatos Exif.

Amazon Rekognition no realiza la corrección de imagen para imágenes en formato.png ni imágenes.jpeg sin información de orientación en los metadatos Exif de la imagen. Las coordenadas del cuadro delimitador no se traducen y representan las ubicaciones de los objetos antes de girar la imagen.

Type: Cadena

Valores válidos: ROTATE\_0 | ROTATE\_90 | ROTATE\_180 | ROTATE\_270

## Errores

AccessDeniedException

No tiene autorización para realizar la acción.

Código de estado HTTP: 400

ImageTooLargeException

La imagen de entrada tamaño supera el límite permitido. Si estás llamando [DetectProtectiveEquipment \(p. 592\)](#), la tamaño o la resolución de la imagen supera el límite permitido. Para obtener más información, consulte [Directrices y cuotas de Amazon Rekognition \(p. 847\)](#).

Código de estado HTTP: 400

InternalServerError

Amazon Lex ha tenido un problema de servicio. Pruebe la llamada de nuevo.

Código de estado HTTP: 500

InvalidImageFormatException

No se admite el formato de imagen proporcionado.

Código de estado HTTP: 400

InvalidParameterException

La entrada parámetro infringió una restricción. Valide el parámetro antes de llamar a la operación de la API de nuevo.

Código de estado HTTP: 400

InvalidS3ObjectException

Amazon Rekognition no puede obtener acceso al objeto de S3 especificado en la solicitud.

Código de estado HTTP: 400

ProvisionedThroughputExceededException

El número de solicitudes ha superado su límite de rendimiento. Si necesita aumentar este límite, póngase en contacto con Amazon Rekognition.

Código de estado HTTP: 400

ThrottlingException

Amazon Lex no puede procesar temporalmente la solicitud. Pruebe la llamada de nuevo.

Código de estado HTTP: 500

## Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- [AWS Command Line Interface](#)
- [SDK de AWS para .NET](#)
- [AWS SDK para C++](#)

- [AWS SDK para Go](#)
- [AWSSDK para Java V2](#)
- [AWS SDK para JavaScript](#)
- [SDK de AWS para PHP V3](#)
- [SDK de AWS para Python](#)
- [SDK de AWS para Ruby V3](#)

## DetectLabels

Detecta las instancias de entidades del mundo real dentro de una imagen (JPEG o PNG) que se proporciona como entrada. Esto incluye objetos como flores, árboles y mesa; eventos como boda, graduación y fiesta de cumpleaños; y conceptos como paisaje, noche y naturaleza.

Para ver un ejemplo, consulte [Análisis de imágenes almacenadas en un bucket de Amazon S3 \(p. 30\)](#).

### Note

`DetectLabels` no admite la detección de actividades. Sin embargo, la detección de actividad se admite para la detección de etiquetas en vídeos. Para obtener más información, consulte [StartLabelDetection \(p. 701\)](#).

Puede pasar la imagen de entrada como bytes de imagen con codificación en base64 o como referencia a una imagen de un bucket de Amazon S3. Si utiliza la AWS CLI para llamar a las operaciones de Amazon Rekognition, no es posible pasar bytes de imágenes. La imagen debe ser un archivo de formato PNG o JPEG.

Para cada objeto, escena y concepto, la API devuelve una o más etiquetas. Cada etiqueta proporciona el nombre del objeto y el nivel de confianza de que la imagen contiene el objeto. Por ejemplo, supongamos que la imagen de entrada tiene un faro, el mar y una roca. La respuesta incluye las tres etiquetas, una para cada objeto.

```
{Name: lighthouse, Confidence: 98.4629}

{Name: rock, Confidence: 79.2097}

{Name: sea, Confidence: 75.061}
```

En el ejemplo anterior, la operación devuelve una etiqueta para cada uno de los tres objetos. La operación también puede devolver varias etiquetas para el mismo objeto de la imagen. Por ejemplo, si la imagen de entrada muestra una flor (por ejemplo, un tulipán), la operación podría devolver las tres etiquetas siguientes.

```
{Name: flower, Confidence: 99.0562}

{Name: plant, Confidence: 99.0562}

{Name: tulip, Confidence: 99.0562}
```

En este ejemplo, el algoritmo de detección identifica con mayor precisión la flor como tulipán.

Como respuesta, la API devuelve una matriz de etiquetas. Además, la respuesta también incluye la corrección de orientación. Si lo desea, puede especificar `MinConfidence` para controlar el umbral de confianza de las etiquetas devueltas. El valor predeterminado es 55%. También puede agregar la `MaxLabels` para limitar el número de etiquetas devueltas.

### Note

Si el objeto detectado es una persona, la operación no proporciona los mismos detalles faciales que el [DetectFaces \(p. 578\)](#) la operación proporciona.

`DetectLabels` devuelve cuadros delimitadores para instancias de etiquetas de objeto comunes en una matriz de [Instance \(p. 789\)](#) objetos. Un `Instance` contiene un [BoundingBox \(p. 740\)](#), para obtener la ubicación de la etiqueta en la imagen. También incluye la confianza por la que se detectó el cuadro delimitador.

`DetectLabels` devuelve también una taxonomía jerárquica de las etiquetas detectadas. Por ejemplo, a un automóvil detectado se le puede asignar la etiqueta coche. La etiqueta coche tiene dos etiquetas principales:

Vehículo(su padre) y Transporte(su abuelo). La respuesta devuelve toda la lista de antepasados de una etiqueta. Cada antepasado es una etiqueta única en la respuesta. En el ejemplo anterior, Coche, Vehículo, y Transporte se devuelven como etiquetas únicas en la respuesta.

Se trata de una operación de API sin estado. Es decir, la operación no persiste ningún dato.

Esta operación requiere permisos para realizar la acción `rekognition:DetectLabels`.

## Sintaxis de la solicitud

```
{
 "Image": {
 "Bytes": blob,
 "S3Object": {
 "Bucket": "string",
 "Name": "string",
 "Version": "string"
 }
 },
 "MaxLabels": number,
 "MinConfidence": number
}
```

## Parámetros de solicitud

La solicitud acepta los siguientes datos en formato JSON.

### [Image \(p. 584\)](#)

La imagen de entrada como bytes codificados en base64 o un objeto S3. Si utiliza la AWS CLI para llamar a las operaciones de Amazon Rekognition, no es posible pasar bytes de imágenes. Las imágenes almacenadas en un bucket de S3 no necesitan estar codificadas en base64.

Si utiliza un SDK de AWS para llamar a Amazon Rekognition, es posible que no tenga que codificar en base 64 bytes de imagen pasados mediante elBytes. Para obtener más información, consulte [Especificaciones de imágenes \(p. 29\)](#).

Tipo: objeto [Image \(p. 786\)](#)

Obligatorio: Sí

### [MaxLabels \(p. 584\)](#)

Número máximo de etiquetas que desea que devuelva el servicio en la respuesta. El servicio devuelve el número especificado de etiquetas de mayor confianza.

Type: Entero

Rango válido: Valor mínimo de 0.

Obligatorio: No

### [MinConfidence \(p. 584\)](#)

Especifica el nivel de confianza mínimo de las etiquetas que se van a devolver. Amazon Rekognition no devuelve ninguna etiqueta con confianza inferior a este valor especificado.

Si MinConfidenceno se especifica, la operación devuelve etiquetas con valores de confianza superiores o iguales al 55 por ciento.

Type: Float

Rango válido: Valor mínimo de 0. Valor máximo de 100.

Obligatorio: No

## Sintaxis de la respuesta

```
{
 "LabelModelVersion": "string",
 "Labels": [
 {
 "Confidence": number,
 "Instances": [
 {
 "BoundingBox": {
 "Height": number,
 "Left": number,
 "Top": number,
 "Width": number
 },
 "Confidence": number
 }
],
 "Name": "string",
 "Parents": [
 {
 "Name": "string"
 }
]
 }
],
 "OrientationCorrection": "string"
}
```

## Elementos de respuesta

Si la acción se realiza correctamente, el servicio devuelve una respuesta HTTP 200.

El servicio devuelve los datos siguientes en formato JSON.

### [LabelModelVersion \(p. 585\)](#)

Número de versión del modelo de detección de etiquetas que se ha utilizado para detectar etiquetas.

Type: Cadena

### [Labels \(p. 585\)](#)

Matriz de etiquetas para los objetos del mundo real detectados.

Type: Matriz de [Label \(p. 793\)](#) objects

### [OrientationCorrection \(p. 585\)](#)

El valor de OrientationCorrection siempre es nulo.

Si la imagen de entrada está en formato .jpeg, puede contener metadatos de formato de archivo de imagen intercambiable (EXIF) que incluyen la orientación de la imagen. Amazon Rekognition utiliza esta información de orientación para realizar la corrección de imagen. Las coordenadas del cuadro delimitador se traducen para representar las ubicaciones de los objetos después de utilizar la información de orientación de los metadatos Exif para corregir la orientación de la imagen. Las imágenes en formato .png no contienen metadatos Exif.

Amazon Rekognition no realiza la corrección de imagen para imágenes en formato.png ni imágenes.jpeg sin información de orientación en los metadatos Exif de la imagen. Las coordenadas del cuadro delimitador no se traducen y representan las ubicaciones de los objetos antes de girar la imagen.

Type: Cadena

Valores válidos: ROTATE\_0 | ROTATE\_90 | ROTATE\_180 | ROTATE\_270

## Errores

AccessDeniedException

No tiene autorización para realizar la acción.

Código de estado HTTP: 400

ImageTooLargeException

La imagen de entrada tamaño supera el límite permitido. Si estás llamando [DetectProtectiveEquipment \(p. 592\)](#), el tamaño o la resolución de la imagen supera el límite permitido. Para obtener más información, consulte [Directrices y cuotas de Amazon Rekognition \(p. 847\)](#).

Código de estado HTTP: 400

InternalServerError

Amazon Lex ha tenido un problema de servicio. Pruebe la llamada de nuevo.

Código de estado HTTP: 500

InvalidImageFormatException

No se admite el formato de imagen proporcionado.

Código de estado HTTP: 400

InvalidParameterException

El parámetro de entrada infringió una restricción. Valide el parámetro antes de llamar a la operación de la API de nuevo.

Código de estado HTTP: 400

InvalidS3ObjectException

Amazon Rekognition no puede obtener acceso al objeto de S3 especificado en la solicitud.

Código de estado HTTP: 400

ProvisionedThroughputExceededException

El número de solicitudes ha superado su límite de rendimiento. Si necesita aumentar este límite, póngase en contacto con Amazon Rekognition.

Código de estado HTTP: 400

ThrottlingException

Amazon Lex no puede procesar temporalmente la solicitud. Pruebe la llamada de nuevo.

Código de estado HTTP: 500

## Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- [AWS Command Line Interface](#)
- [SDK de AWS para .NET](#)
- [AWS SDK para C++](#)
- [AWS SDK para Go](#)
- [AWSSDK para Java V2](#)
- [AWS SDK para JavaScript](#)
- [SDK de AWS para PHP V3](#)
- [SDK de AWS para Python](#)
- [SDK de AWS para Ruby V3](#)

## DetectModerationLabels

Detecta contenido inseguro en una imagen en formato JPEG o PNG especificada.

Usar `DetectModerationLabels` para moderar las imágenes en función de sus necesidades. Por ejemplo, es posible que desee filtrar imágenes que contienen desnudos, pero no imágenes que contengan contenido sugerente.

Para filtrar imágenes, utilice las etiquetas devueltas por `DetectModerationLabels` para determinar qué tipos de contenido son adecuados.

Para obtener más información sobre las etiquetas de moderación, consulte [Moderación de contenido \(p. 298\)](#). Para obtener una lista de etiquetas de moderación en Amazon Rekognition, consulte [Uso de las API de moderación de imagen y vídeo](#).

Se pasa la imagen de entrada como bytes de imagen codificados en base64 o como referencia a una imagen de un bucket de Amazon S3. Si utiliza la CLI de AWS para llamar a las operaciones de Amazon Rekognition, no es posible pasar bytes de imágenes. La imagen debe ser un archivo de formato PNG o JPEG.

## Sintaxis de la solicitud

```
{
 "HumanLoopConfig": {
 "DataAttributes": {
 "ContentClassifiers": ["string"]
 },
 "FlowDefinitionArn": "string",
 "HumanLoopName": "string"
 },
 "Image": {
 "Bytes": blob,
 "S3Object": {
 "Bucket": "string",
 "Name": "string",
 "Version": "string"
 }
 },
 "MinConfidence": number
}
```

## Parámetros de solicitud

La solicitud acepta los siguientes datos en formato JSON.

### [HumanLoopConfig \(p. 588\)](#)

Configura la configuración para la evaluación humana, incluida la FlowDefinition a la que se enviará la imagen.

Tipo: objeto [HumanLoopConfig \(p. 784\)](#)

Obligatorio: No

### [Image \(p. 588\)](#)

La imagen de entrada como bytes codificados en base64 o un objeto S3. Si utiliza la CLI de AWS para llamar a las operaciones de Amazon Rekognition, no es posible pasar bytes de imágenes codificados en base64.

Si utiliza un SDK de AWS para llamar a Amazon Rekognition, es posible que no tenga que codificar en base 64 bytes de imagen pasados mediante laBytes. Para obtener más información, consulte [Especificaciones de imágenes \(p. 29\)](#).

Tipo: objeto [Image \(p. 786\)](#)

Obligatorio: Sí

[MinConfidence \(p. 588\)](#)

Especifica el nivel de confianza mínimo de las etiquetas que se van a devolver. Amazon Rekognition no devuelve ninguna etiqueta con un nivel de confianza inferior al valor especificado.

Si no especifica [MinConfidence](#), la operación devuelve etiquetas con valores de confianza superiores o iguales al 50 por ciento.

Type: Float

Rango válido: Valor mínimo de 0. Valor máximo de 100.

Obligatorio: No

## Sintaxis de la respuesta

```
{
 "HumanLoopActivationOutput": {
 "HumanLoopActivationConditionsEvaluationResults": "string",
 "HumanLoopActivationReasons": ["string"],
 "HumanLoopArn": "string"
 },
 "ModerationLabels": [
 {
 "Confidence": number,
 "Name": "string",
 "ParentName": "string"
 }
],
 "ModerationModelVersion": "string"
}
```

## Elementos de respuesta

Si la acción se realiza correctamente, el servicio devuelve una respuesta HTTP 200.

El servicio devuelve los datos siguientes en formato JSON.

[HumanLoopActivationOutput \(p. 589\)](#)

Muestra los resultados de la evaluación humana en bucle.

Tipo: objeto [HumanLoopActivationOutput \(p. 783\)](#)

[ModerationLabels \(p. 589\)](#)

Matriz de etiquetas de moderación detectadas y el tiempo, en milisegundos desde el principio del vídeo, cuando se detectó.

Type: Matriz de [ModerationLabel \(p. 796\)](#) objects

[ModerationModelVersion \(p. 589\)](#)

Número de versión del modelo de detección de moderación que se utilizó para detectar contenido no seguro.

Type: Cadena

## Errores

AccessDeniedException

No tiene autorización para realizar la acción.

Código de estado HTTP: 400

HumanLoopQuotaExceededException

El número de revisiones humanas en curso que has superado el número permitido.

Código de estado HTTP: 400

ImageTooLargeException

La imagen de entrada tamaño supera el límite permitido. Si estás llamando [DetectProtectiveEquipment \(p. 592\)](#), el tamaño o la resolución de imágenes superan el límite permitido. Para obtener más información, consulte [Directrices y cuotas de Amazon Rekognition \(p. 847\)](#).

Código de estado HTTP: 400

InternalServerError

Amazon Lex ha tenido un problema de servicio. Pruebe la llamada de nuevo.

Código de estado HTTP: 500

InvalidImageFormatException

No se admite el formato de imagen proporcionado.

Código de estado HTTP: 400

InvalidParameterException

El parámetro de entrada infringió una restricción. Valide el parámetro antes de llamar a la operación de la API de nuevo.

Código de estado HTTP: 400

InvalidS3ObjectException

Amazon Rekognition no puede obtener acceso al objeto de S3 especificado en la solicitud.

Código de estado HTTP: 400

ProvisionedThroughputExceededException

El número de solicitudes ha superado su límite de rendimiento. Si necesita aumentar este límite, póngase en contacto con Amazon Rekognition.

Código de estado HTTP: 400

ThrottlingException

Amazon Lex no puede procesar temporalmente la solicitud. Pruebe la llamada de nuevo.

Código de estado HTTP: 500

## Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- [AWS Command Line Interface](#)
- [SDK de AWS para .NET](#)
- [AWS SDK para C++](#)
- [AWS SDK para Go](#)
- [AWSSDK para Java V2](#)
- [AWS SDK para JavaScript](#)
- [SDK de AWS para PHP V3](#)
- [SDK de AWS para Python](#)
- [SDK de AWS para Ruby V3](#)

## DetectProtectiveEquipment

Detecta el equipo de protección personal (EPP) usado por las personas detectadas en una imagen. Amazon Rekognition puede detectar los siguientes tipos de EPP.

- Funda para rostros
- Funda de mano
- Head cover

La imagen de entrada se transfiere como bytes de imagen cifrados en base64 o como referencia a una imagen de un bucket de Amazon S3. La imagen debe ser un archivo de formato PNG o JPG.

`DetectProtectiveEquipment` detecta EPP usado por hasta 15 personas detectadas en una imagen.

Para cada persona detectada en la imagen, la API devuelve un conjunto de partes del cuerpo (cara, cabeza, izquierda, mano derecha). Para cada parte del cuerpo, se devuelve una matriz de elementos de EPP detectados, incluido un indicador de si el EPP cubre o no la parte del cuerpo. La API devuelve la confianza que tiene en cada detección (cobertura de persona, EPP, parte del cuerpo y parte del cuerpo). También devuelve un cuadro delimitador ([BoundingBox \(p. 740\)](#)) para cada persona detectada y cada elemento de EPP detectado.

Si lo desea, puede solicitar un resumen de los elementos de EPP detectados con `SummarizationAttributes` parámetro de entrada. El resumen proporciona la siguiente información.

- Las personas detectadas que llevan todos los tipos de EPP especificados.
- Las personas detectadas que no llevan todos los tipos de EPP especificados.
- Las personas detectadas en las que no se pudo determinar el adorno de EPP.

Se trata de una operación de API sin estado. Es decir, la operación no persiste ningún dato.

Esta operación requiere permisos para realizar la acción `rekognition:DetectProtectiveEquipment`.

## Sintaxis de la solicitud

```
{
 "Image": {
 "Bytes": blob,
 "S3Object": {
 "Bucket": "string",
 "Name": "string",
 "Version": "string"
 }
 },
 "SummarizationAttributes": {
 "MinConfidence": number,
 "RequiredEquipmentTypes": ["string"]
 }
}
```

## Parámetros de solicitud

La solicitud acepta los siguientes datos en formato JSON.

### [Image \(p. 592\)](#)

Imagen en la que desea detectar EPP en personas detectadas. La imagen se puede transferir como bytes de imagen o hacer referencia a una imagen almacenada en un bucket de Amazon S3.

Tipo: objeto [Image \(p. 786\)](#)

Obligatorio: Sí

### [SummarizationAttributes \(p. 592\)](#)

Matriz de tipos de EPP que desea resumir.

Tipo: objeto [ProtectiveEquipmentSummarizationAttributes \(p. 813\)](#)

Obligatorio: No

## Sintaxis de la respuesta

```
{
 "Persons": [
 {
 "BodyParts": [
 {
 "Confidence": number,
 "EquipmentDetections": [
 {
 "BoundingBox": {
 "Height": number,
 "Left": number,
 "Top": number,
 "Width": number
 },
 "Confidence": number,
 "CoversBodyPart": {
 "Confidence": number,
 "Value": boolean
 },
 "Type": "string"
 }
],
 "Name": "string"
 }
],
 "BoundingBox": {
 "Height": number,
 "Left": number,
 "Top": number,
 "Width": number
 },
 "Confidence": number,
 "Id": number
 }
],
 "ProtectiveEquipmentModelVersion": "string",
 "Summary": {
 "PersonsIndeterminate": [number],
 "PersonsWithoutRequiredEquipment": [number],
 "PersonsWithRequiredEquipment": [number]
 }
}
```

## Elementos de respuesta

Si la acción se realiza correctamente, el servicio devuelve una respuesta HTTP 200.

El servicio devuelve los datos siguientes en formato JSON.

### [Persons \(p. 593\)](#)

Una serie de personas detectadas en la imagen (incluidas las personas que no usan EPP).

Type: Matriz de[ProtectiveEquipmentPerson \(p. 812\)](#)objects

### [ProtectiveEquipmentModelVersion \(p. 593\)](#)

Número de versión del modelo de detección de EPP que se ha utilizado para detectar EPP en la imagen.

Type: Cadena

### [Summary \(p. 593\)](#)

Información resumida de los tipos de EPP especificados en elSummarizationAttributes parámetro de entrada.

Tipo: objeto [ProtectiveEquipmentSummary \(p. 814\)](#)

## Errores

### [AccessDeniedException](#)

No tiene autorización para realizar la acción.

Código de estado HTTP: 400

### [ImageTooLargeException](#)

La imagen de entrada tamaño supera el límite permitido. Si estás llamando[DetectProtectiveEquipment \(p. 592\)](#), el tamaño o la resolución de la imagen supera el límite permitido. Para obtener más información, consulte [Directrices y cuotas de Amazon Rekognition \(p. 847\)](#).

Código de estado HTTP: 400

### [InternalServerError](#)

Amazon Lex ha tenido un problema de servicio. Pruebe la llamada de nuevo.

Código de estado HTTP: 500

### [InvalidImageFormatException](#)

No se admite el formato de imagen proporcionado.

Código de estado HTTP: 400

### [InvalidParameterException](#)

El parámetro de entrada infringió una restricción. Valide el parámetro antes de llamar a la operación de la API de nuevo.

Código de estado HTTP: 400

### [InvalidS3ObjectException](#)

Amazon Rekognition no puede obtener acceso al objeto de S3 especificado en la solicitud.

Código de estado HTTP: 400

ProvisionedThroughputExceededException

El número de solicitudes ha superado su límite de rendimiento. Si necesita aumentar este límite, póngase en contacto con Amazon Rekognition.

Código de estado HTTP: 400

ThrottlingException

Amazon Lex no puede procesar temporalmente la solicitud. Pruebe la llamada de nuevo.

Código de estado HTTP: 500

## Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- [AWS Command Line Interface](#)
- [SDK de AWS para .NET](#)
- [AWS SDK para C++](#)
- [AWS SDK para Go](#)
- [AWSSDK para Java V2](#)
- [AWS SDK para JavaScript](#)
- [SDK de AWS para PHP V3](#)
- [SDK de AWS para Python](#)
- [SDK de AWS para Ruby V3](#)

## DetectText

Detecta texto en la imagen de entrada y lo convierte en texto legible por una máquina.

Pasa la imagen de entrada como bytes de imagen con codificación en base64 o como referencia a una imagen de un bucket de Amazon S3. Si utiliza la CLI de AWS para llamar a las operaciones de Amazon Rekognition, debe pasarla como referencia a una imagen de un bucket de Amazon S3. Para la CLI de AWS, no es posible pasar bytes de imágenes. La imagen debe ser un archivo de formato .png o .jpeg.

La `DetectText` operación devuelve texto en una matriz `TextDetections`. Cada `TextDetection` proporciona información sobre una sola palabra o línea de texto detectada en la imagen.

Una palabra consiste en uno o varios caracteres de guión que no están separados por espacios. `DetectText` puede detectar hasta 100 palabras en una imagen.

Una línea es una cadena de palabras equidistantes. Una línea no es necesariamente una frase completa. Por ejemplo, un número de matrícula se detecta como una línea. Una línea finaliza cuando no hay texto alineado después de la misma. Además, una línea finaliza cuando existe un hueco grande entre las palabras, en relación con la longitud de las mismas. Esto significa, en función del hueco entre palabras, que Amazon Rekognition podría detectar varias líneas de texto alineado en la misma dirección. Los puntos no representan el final de una línea. Si una frase abarca varias líneas, la operación `DetectText` devuelve varias líneas.

Para determinar si un `TextDetectionElement` es una línea de texto o una palabra, utilice la `Type` propiedad.

Para detectarlo, el texto debe estar dentro de una orientación de +/- 90° grados con respecto al eje horizontal.

Para obtener más información, consulte [Detección de texto \(p. 315\)](#).

## Sintaxis de la solicitud

```
{
 "Filters": {
 "RegionsOfInterest": [
 {
 "BoundingBox": {
 "Height": number,
 "Left": number,
 "Top": number,
 "Width": number
 }
 }
],
 "WordFilter": {
 "MinBoundingBoxHeight": number,
 "MinBoundingBoxWidth": number,
 "MinConfidence": number
 }
 },
 "Image": {
 "Bytes": blob,
 "S3Object": {
 "Bucket": "string",
 "Name": "string",
 "Version": "string"
 }
 }
}
```

}

## Parámetros de solicitud

La solicitud acepta los siguientes datos en formato JSON.

### [Filters \(p. 596\)](#)

Parámetros opcionales que permiten establecer los criterios que debe cumplir el texto para incluirlo en la respuesta.

Tipo: objeto [DetectTextFilters \(p. 764\)](#)

Obligatorio: No

### [Image \(p. 596\)](#)

La imagen de entrada como bytes codificados en base64 o un objeto Amazon S3. Si utiliza la CLI de AWS para llamar a las operaciones de Amazon Rekognition, no puede pasar bytes de imagen.

Si utiliza un SDK de AWS para llamar a Amazon Rekognition, es posible que no tenga que codificar en base 64 bytes de imagen pasados mediante elBytes. Para obtener más información, consulte [Especificaciones de imágenes \(p. 29\)](#).

Tipo: objeto [Image \(p. 786\)](#)

Obligatorio: Sí

## Sintaxis de la respuesta

```
{
 "TextDetections": [
 {
 "Confidence": number,
 "DetectedText": "string",
 "Geometry": {
 "BoundingBox": {
 "Height": number,
 "Left": number,
 "Top": number,
 "Width": number
 },
 "Polygon": [
 {
 "X": number,
 "Y": number
 }
]
 },
 "Id": number,
 "ParentId": number,
 "Type": "string"
 }
],
 "TextModelVersion": "string"
}
```

## Elementos de respuesta

Si la acción se realiza correctamente, el servicio devuelve una respuesta HTTP 200.

El servicio devuelve los datos siguientes en formato JSON.

[TextDetections \(p. 597\)](#)

Matriz de texto que se ha detectado en la imagen de entrada.

Type: Matriz de[TextDetection \(p. 837\)](#)objects

[TextModelVersion \(p. 597\)](#)

Versión de modelo utilizada para detectar texto.

Type: Cadena

## Errores

[AccessDeniedException](#)

No tiene autorización para realizar la acción.

Código de estado HTTP: 400

[ImageTooLargeException](#)

La imagen de entrada tamaño supera el límite permitido. Si estás llamando[DetectProtectiveEquipment \(p. 592\)](#), la resolución o tamaño de la imagen supera el límite permitido. Para obtener más información, consulte [Directrices y cuotas de Amazon Rekognition \(p. 847\)](#).

Código de estado HTTP: 400

[InternalServerError](#)

Amazon Lex ha tenido un problema de servicio. Pruebe la llamada de nuevo.

Código de estado HTTP: 500

[InvalidImageFormatException](#)

No se admite el formato de imagen proporcionado.

Código de estado HTTP: 400

[InvalidParameterException](#)

El parámetro de entrada infringió una restricción. Valide el parámetro antes de llamar a la operación de la API de nuevo.

Código de estado HTTP: 400

[InvalidS3ObjectException](#)

Amazon Rekognition no puede obtener acceso al objeto de S3 especificado en la solicitud.

Código de estado HTTP: 400

[ProvisionedThroughputExceededException](#)

El número de solicitudes ha superado su límite de rendimiento. Si necesita aumentar este límite, póngase en contacto con Amazon Rekognition.

Código de estado HTTP: 400

[ThrottlingException](#)

Amazon Lex no puede procesar temporalmente la solicitud. Pruebe la llamada de nuevo.

Código de estado HTTP: 500

## Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- [AWS Command Line Interface](#)
- [SDK de AWS para .NET](#)
- [AWS SDK para C++](#)
- [AWS SDK para Go](#)
- [AWSSDK para Java V2](#)
- [AWS SDK para JavaScript](#)
- [SDK de AWS para PHP V3](#)
- [SDK de AWS para Python](#)
- [SDK de AWS para Ruby V3](#)

## DistributeDatasetEntries

Distribuye las entradas (imágenes) de un conjunto de datos de formación en todo el conjunto de datos de formación y el conjunto de datos de prueba de un proyecto. `DistributeDatasetEntries` mueve el 20% de las imágenes del conjunto de datos de entrenamiento al conjunto de datos de prueba. Una entrada es una línea JSON que describe una imagen.

Proporciona los nombres de recursos de Amazon (ARN) del conjunto de datos de formación y del conjunto de datos de prueba de un proyecto. El conjunto de datos de formación debe contener las imágenes que deseé dividir. El conjunto de datos de prueba debe estar vacío. Los conjuntos de datos deben pertenecer al mismo proyecto. Para crear conjuntos de datos de formación y pruebas para un proyecto, llame [CreateDataset \(p. 525\)](#).

La distribución de un conjunto de datos tarda un tiempo en completarse. Para comprobar la llamada de `estadoDescribeDataset`. La operación finaliza cuando el valor `deStatus` para el conjunto de datos de formación y el conjunto de datos de prueba es `UPDATE_COMPLETE`. Si falla la división del conjunto de datos, el valor `deStatus` es `UPDATE_FAILED`.

Esta operación requiere permisos para realizar la acción `rekognition:DistributeDatasetEntries`.

### Sintaxis de la solicitud

```
{
 "Datasets": [
 {
 "Arn": "string"
 }
]
}
```

### Parámetros de solicitud

La solicitud acepta los siguientes datos en formato JSON.

#### Datasets (p. 600)

El ARNs del conjunto de datos de formación y el conjunto de datos de prueba que desea utilizar. Los conjuntos de datos deben pertenecer al mismo proyecto. El conjunto de datos de prueba debe estar vacío.

Type: Matriz de [DistributeDataset \(p. 765\)](#) objects

Miembros de la matriz: Número fijo de 2 elementos.

Obligatorio: Sí

### Elementos de respuesta

Si la acción se realiza correctamente, el servicio devuelve una respuesta HTTP 200 con un cuerpo HTTP vacío.

### Errores

#### AccessDeniedException

No tiene autorización para realizar la acción.

Código de estado HTTP: 400

InternalServerError

Amazon Lex ha tenido un problema de servicio. Pruebe la llamada de nuevo.

Código de estado HTTP: 500

InvalidParameterException

El parámetro de entrada infringió una restricción. Valide el parámetro antes de llamar a la operación de la API de nuevo.

Código de estado HTTP: 400

ProvisionedThroughputExceededException

El número de solicitudes ha superado su límite de rendimiento. Si necesita aumentar este límite, póngase en contacto con Amazon Rekognition.

Código de estado HTTP: 400

ResourceNotFoundException

No se encuentra el recurso especificado en la solicitud.

Código de estado HTTP: 400

ResourceNotReadyException

El recurso solicitado no está listo. Por ejemplo, esta excepción se produce cuando `llamaDetectCustomLabels` con una versión de modelo que no está implementada.

Código de estado HTTP: 400

ThrottlingException

Amazon Lex no puede procesar temporalmente la solicitud. Pruebe la llamada de nuevo.

Código de estado HTTP: 500

## Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- [AWS Command Line Interface](#)
- [SDK de AWS para .NET](#)
- [AWS SDK para C++](#)
- [AWS SDK para Go](#)
- [AWSSDK para Java V2](#)
- [AWS SDK para JavaScript](#)
- [SDK de AWS para PHP V3](#)
- [SDK de AWS para Python](#)
- [SDK de AWS para Ruby V3](#)

## GetCelebrityInfo

Obtiene el nombre e información adicional acerca de un famoso en función de su Amazon Rekognition ID. La información adicional se devuelve como una matriz de URL. Si no hay información adicional sobre la celebridad, esta lista está vacía.

Para obtener más información, consulte [Obtención de información sobre un famoso \(p. 294\)](#).

Esta operación requiere permisos para realizar la acción `rekognition:GetCelebrityInfo`.

### Sintaxis de la solicitud

```
{
 "Id": "string"
}
```

### Parámetros de solicitud

La solicitud acepta los siguientes datos en formato JSON.

#### [Id \(p. 602\)](#)

La identificación de la celebridad. Obtiene la identificación de celebridades de una llamada [alRecognizeCelebrities \(p. 671\)](#)Operación, que reconoce famosos en una imagen.

Type: Cadena

Patrón: [ 0-9A-Za-z ]\*

Obligatorio: Sí

### Sintaxis de la respuesta

```
{
 "KnownGender": {
 "Type": "string"
 },
 "Name": "string",
 "Urls": ["string"]
}
```

### Elementos de respuesta

Si la acción se realiza correctamente, el servicio devuelve una respuesta HTTP 200.

El servicio devuelve los datos siguientes en formato JSON.

#### [KnownGender \(p. 602\)](#)

Recupera el género conocido de la celebridad.

Tipo: objeto [KnownGender \(p. 792\)](#)

#### [Name \(p. 602\)](#)

El nombre de la celebridad.

Type: Cadena

[Urls \(p. 602\)](#)

Una serie de URL que apuntan a información adicional de celebridades.

Type: Matriz de cadenas

Miembros de matriz: El número mínimo es 0 elementos. Número máximo de 255 elementos.

## Errores

[AccessDeniedException](#)

No tiene autorización para realizar la acción.

Código de estado HTTP: 400

[InternalServerError](#)

Amazon Lex ha tenido un problema de servicio. Pruebe la llamada de nuevo.

Código de estado HTTP: 500

[InvalidParameterException](#)

El parámetro de entrada infringió una restricción. Valide el parámetro antes de llamar a la operación de la API de nuevo.

Código de estado HTTP: 400

[ProvisionedThroughputExceededException](#)

El número de solicitudes ha superado su límite de rendimiento. Si necesita aumentar este límite, póngase en contacto con Amazon Rekognition.

Código de estado HTTP: 400

[ResourceNotFoundException](#)

El recurso especificado en la solicitud no se encuentra.

Código de estado HTTP: 400

[ThrottlingException](#)

Amazon Lex no puede procesar temporalmente la solicitud. Pruebe la llamada de nuevo.

Código de estado HTTP: 500

## Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- [AWS Command Line Interface](#)
- [SDK de AWS para .NET](#)
- [AWS SDK para C++](#)
- [AWS SDK para Go](#)
- [AWSSDK para Java V2](#)
- [AWS SDK para JavaScript](#)

- [SDK de AWS para PHP V3](#)
- [SDK de AWS para Python](#)
- [SDK de AWS para Ruby V3](#)

## GetCelebrityRecognition

Obtiene resultados de reconocimiento de famosos para un Amazon Rekognition Video iniciado por [StartCelebrityRecognition \(p. 685\)](#).

El reconocimiento de famosos de un vídeo es una operación asíncrona. El análisis se inicia mediante una llamada a [StartCelebrityRecognition \(p. 685\)](#) que devuelve un identificador de trabajo (`JobId`).

Cuando finaliza la operación de reconocimiento de celebridades, Amazon Rekognition Video publica un estado de finalización en el tema Amazon Simple Notification Service registrado en la llamada inicial a `StartCelebrityRecognition`. Para obtener los resultados del análisis de reconocimiento de celebridades, primero compruebe que el valor de estado publicado en el tema de Amazon SNS es `SUCCEEDED`. Si es así, llame `GetCelebrityDetection` y pase el identificador de trabajo (`JobId`) desde la llamada inicial hasta `startCelebrityDetection`.

Para obtener más información, consulte [Trabajar con vídeos almacenados \(p. 63\)](#).

`GetCelebrityRecognition` devuelve famosos detectados y los momentos en que se detectan en una matriz (`Celebrities`) de [CelebrityRecognition \(p. 746\)](#) objetos. Cada `CelebrityRecognition` contiene información sobre la celebridad de [CelebrityDetail \(p. 744\)](#) objeto y el tiempo, `Timestamp`, se detectó a la celebridad. Este [CelebrityDetail \(p. 744\)](#) almacena información sobre los atributos de cara de la celebridad detectada, un cuadro delimitador de caras, género conocido, nombre de la celebridad y una estimación de confianza.

### Note

`GetCelebrityRecognition` solo devuelve los atributos faciales predeterminados (`BoundingBox`, `Confidence`, `Landmarks`, `Pose`, y `Quality`). La `BoundingBox` solo se aplica a la instancia de cara detectada. Los demás atributos faciales enumerados en el `Face` no se devuelven el objeto de la siguiente sintaxis de respuesta. Para obtener más información, consulte [FaceDetail \(p. 773\)](#).

Por defecto, `Celebrities` la matriz se ordena por tiempo (milisegundos desde el comienzo del vídeo). También puedes ordenar la matriz por celebridades especificando el valor `IDen` la `SortBy` parámetro de entrada.

La `CelebrityDetail` incluye el identificador de celebridades y URL de información adicional. Si no almacenas las URL de información adicional, puedes obtenerlas más tarde llamando [GetCelebrityInfo \(p. 602\)](#) con el identificador de celebridades.

No se devuelve información sobre rostros no reconocidos como celebridades.

Utilice el parámetro `MaxResults` para limitar el número de etiquetas devueltas. Si hay más resultados de los especificados en `MaxResults`, el valor de `NextToken` de la respuesta de la operación contiene un token de paginación para obtener el siguiente conjunto de resultados. Para obtener la siguiente página de resultados, llame `GetCelebrityDetection` y rellénela `NextToken` parámetro de solicitud con el valor de token devuelto desde la llamada anterior a `GetCelebrityRecognition`.

## Sintaxis de la solicitud

```
{
 "JobId": "string",
 "MaxResults": number,
 "NextToken": "string",
 "SortBy": "string"
}
```

## Parámetros de solicitud

La solicitud acepta los siguientes datos en formato JSON.

### [JobId \(p. 605\)](#)

Identificador de Job para el análisis de reconocimiento de celebridades requerido. Puede obtener el identificador del trabajo de una llamada a `StartCelebrityRecognition`.

Type: Cadena

Restricciones de longitud: Longitud mínima de 1. La longitud máxima es 64.

Patrón: ^[a-zA-Z0-9-\_]+\$

Obligatorio: Sí

### [MaxResults \(p. 605\)](#)

Número máximo de resultados que devolver por llamada paginada. El valor más grande que puede especificar es 1000. Si especifica un valor superior a 1 000, se devolverá un máximo de 1 000 resultados. El valor predeterminado es 1000.

Type: Entero

Rango válido: Valor mínimo de 1.

Obligatorio: No

### [NextToken \(p. 605\)](#)

Si la respuesta anterior estaba incompleta (porque hay más celebridades reconocidas que recuperar), Amazon Rekognition Video devuelve un token de paginación en la respuesta. Puedes utilizar este token de paginación para recuperar el siguiente grupo de famosos.

Type: Cadena

Restricciones de longitud: La longitud máxima es de 255 caracteres.

Obligatorio: No

### [SortBy \(p. 605\)](#)

Ordenar para usar para celebridades devueltas en `Celebrities`. Especifique `.ID` para ordenar por el identificador de celebridades, especifique `TIMESTAMP` para ordenar por el momento en que se reconoció a la celebridad.

Type: Cadena

Valores válidos: `ID` | `TIMESTAMP`

Obligatorio: No

## Sintaxis de la respuesta

```
{
 "Celebrities": [
 {
 "Celebrity": {
 "BoundingBox": {
 "Height": number,
```

```
"Left": number,
"Top": number,
"Width": number
},
"Confidence": number,
"Face": {
 "AgeRange": {
 "High": number,
 "Low": number
 },
 "Beard": {
 "Confidence": number,
 "Value": boolean
 },
 "BoundingBox": {
 "Height": number,
 "Left": number,
 "Top": number,
 "Width": number
 },
 "Confidence": number,
 "Emotions": [
 {
 "Confidence": number,
 "Type": string
 }
],
 "Eyeglasses": {
 "Confidence": number,
 "Value": boolean
 },
 "EyesOpen": {
 "Confidence": number,
 "Value": boolean
 },
 "Gender": {
 "Confidence": number,
 "Value": string
 },
 "Landmarks": [
 {
 "Type": string,
 "X": number,
 "Y": number
 }
],
 "MouthOpen": {
 "Confidence": number,
 "Value": boolean
 },
 "Mustache": {
 "Confidence": number,
 "Value": boolean
 },
 "Pose": {
 "Pitch": number,
 "Roll": number,
 "Yaw": number
 },
 "Quality": {
 "Brightness": number,
 "Sharpness": number
 },
 "Smile": {
 "Confidence": number,
 "Value": boolean
 }
},
```

```
 },
 "Sunglasses": {
 "Confidence": number,
 "Value": boolean
 }
 },
 "Id": "string",
 "KnownGender": {
 "Type": "string"
 },
 "Name": "string",
 "Urls": ["string"]
},
"Timestamp": number
],
"JobStatus": "string",
"NextToken": "string",
"StatusMessage": "string",
"VideoMetadata": {
 "Codec": "string",
 "ColorRange": "string",
 "DurationMillis": number,
 "Format": "string",
 "FrameHeight": number,
 "FrameRate": number,
 "FrameWidth": number
}
}
```

## Elementos de respuesta

Si la acción se realiza correctamente, el servicio devuelve una respuesta HTTP 200.

El servicio devuelve los datos siguientes en formato JSON.

### Celebrities (p. 606)

Variedad de famosos reconocidos en el vídeo.

Type: Matriz de[CelebrityRecognition \(p. 746\)](#)objects

### JobStatus (p. 606)

El estado actual del trabajo de reconocimiento de famosos.

Type: Cadena

Valores válidos: IN\_PROGRESS | SUCCEEDED | FAILED

### NextToken (p. 606)

Si la respuesta es truncada, Amazon Rekognition Video devuelve este token que puede utilizar en la solicitud posterior para recuperar el siguiente grupo de famosos.

Type: Cadena

Restricciones de longitud: La longitud máxima es de 255 caracteres.

### StatusMessage (p. 606)

Si el trabajo falla, StatusMessage proporciona un mensaje de error descriptivo.

Type: Cadena

## [VideoMetadata \(p. 606\)](#)

Información sobre un vídeo que Amazon Rekognition Video analizó. `videometadatase` se devuelve en cada página de respuestas paginadas de una operación de Amazon Rekognition Video.

Tipo: objeto [VideoMetadata \(p. 845\)](#)

## Errores

### AccessDeniedException

No tiene autorización para realizar la acción.

Código de estado HTTP: 400

### InternalServerError

Amazon Lex ha tenido un problema de servicio. Pruebe la llamada de nuevo.

Código de estado HTTP: 500

### InvalidPaginationTokenException

El token de paginación de la solicitud no es válido.

Código de estado HTTP: 400

### InvalidParameterException

El parámetro de entrada infringió una restricción. Valide el parámetro antes de llamar a la operación de la API de nuevo.

Código de estado HTTP: 400

### ProvisionedThroughputExceededException

El número de solicitudes ha superado su límite de rendimiento. Si necesita aumentar este límite, póngase en contacto con Amazon Rekognition.

Código de estado HTTP: 400

### ResourceNotFoundException

No se encuentra el recurso especificado en la solicitud.

Código de estado HTTP: 400

### ThrottlingException

Amazon Lex no puede procesar temporalmente la solicitud. Pruebe la llamada de nuevo.

Código de estado HTTP: 500

## Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- [AWS Command Line Interface](#)
- [SDK de AWS para .NET](#)
- [AWS SDK para C++](#)
- [AWS SDK para Go](#)

- [AWSSDK para Java V2](#)
- [AWS SDK para JavaScript](#)
- [SDK de AWS para PHP V3](#)
- [SDK de AWS para Python](#)
- [SDK de AWS para Ruby V3](#)

## GetContentModeration

Obtiene resultados de análisis de contenido inapropiados, no deseados u ofensivos para un análisis de Amazon Rekognition Video iniciado por[StartContentModeration \(p. 689\)](#). Para obtener una lista de etiquetas de moderación en Amazon Rekognition, consulte[Uso de las API de moderación de imagen y vídeo](#).

Amazon Rekognition Video La detección de contenido inapropiado u ofensiva en un vídeo almacenado es una operación asíncrona. Comienza el análisis llamando[StartContentModeration \(p. 689\)](#)que devuelve un identificador de trabajo (`JobId`). Cuando finaliza el análisis, Amazon Rekognition Video publica un estado de finalización en el tema Amazon Simple Notification Service registrado en la llamada inicial `aStartContentModeration`. Para obtener los resultados del análisis de contenido, primero compruebe que el valor de estado publicado en el tema de Amazon SNS es `SUCCEEDED`. Si es así, llame[GetContentModeration](#) y pase el identificador de trabajo (`JobId`) desde la llamada inicial hasta `aStartContentModeration`.

Para obtener más información, consulte [Trabajar con vídeos almacenados \(p. 63\)](#).

`GetContentModeration`devuelve etiquetas de moderación de contenido detectadas inapropiadas, no deseadas u ofensivas y el momento en que se detectan, en una matriz,`ModerationLabels`, de[ContentModerationDetection \(p. 751\)](#)objetos.

De forma predeterminada, las etiquetas moderadas se devuelven ordenadas por tiempo, en milisegundos desde el inicio del vídeo. También puede ordenarlos por etiqueta moderada especificando`NAME`para la`SortBy`parámetro de entrada.

Dado que el análisis de vídeo puede devolver un gran número de resultados, utilice el`MaxResults`para limitar el número de etiquetas devueltas en una sola llamada `aGetContentModeration`. Si hay más resultados de los especificados en`MaxResults`, el valor de`NextToken`en la respuesta de la operación contiene un token de paginación para obtener el siguiente grupo de resultados. Para obtener la siguiente página de resultados, llame[GetContentModeration](#) y rellénela`NextToken`parámetro de solicitud con el valor de`NextToken`devuelto de la llamada anterior `aGetContentModeration`.

Para obtener más información, consulte [Moderación de contenido \(p. 298\)](#).

## Sintaxis de la solicitud

```
{
 "JobId": "string",
 "MaxResults": number,
 "NextToken": "string",
 "SortBy": "string"
}
```

## Parámetros de solicitud

La solicitud acepta los siguientes datos en formato JSON.

### `JobId` (p. 611)

Identificador del trabajo de moderación de contenido inapropiado, no deseado u ofensivo. Usar`JobId`para identificar el trabajo en una llamada posterior `aGetContentModeration`.

Type: Cadena

Restricciones de longitud: Longitud mínima de 1. La longitud máxima es 64.

Patrón: ^[a-zA-Z0-9-\_]+\$

Obligatorio: Sí

#### [MaxResults \(p. 611\)](#)

Número máximo de resultados a devolver por llamada paginada. El valor más grande que puede especificar es 1000. Si especifica un valor superior a 1 000, se devolverá un máximo de 1 000 resultados. El valor predeterminado es 1000.

Type: Entero

Rango válido: Valor mínimo de 1.

Obligatorio: No

#### [NextToken \(p. 611\)](#)

Si la respuesta anterior estaba incompleta (porque hay más datos que recuperar), Amazon Rekognition devuelve un token de paginación en la respuesta. Puede utilizar este token de paginación para recuperar el siguiente grupo de etiquetas de moderación de contenido.

Type: Cadena

Restricciones de longitud: La longitud máxima es de 255 caracteres.

Obligatorio: No

#### [SortBy \(p. 611\)](#)

Ordenar para utilizar para los elementos de laModerationLabelDetections matriz.

Usar `TIMESTAMP` para ordenar los elementos de matriz por las etiquetas de tiempo que se detectan.

Usar `NAME` para agrupar alfabéticamente elementos de una etiqueta juntos. Dentro de cada grupo de etiquetas, el elemento de matriz se ordena por confianza de detección. La ordenación predeterminada es por `TIMESTAMP`.

Type: Cadena

Valores válidos: `NAME` | `TIMESTAMP`

Obligatorio: No

## Sintaxis de la respuesta

```
{
 "JobStatus": "string",
 "ModerationLabels": [
 {
 "ModerationLabel": {
 "Confidence": number,
 "Name": "string",
 "ParentName": "string"
 },
 "Timestamp": number
 }
],
 "ModerationModelVersion": "string",
 "NextToken": "string",
 "StatusMessage": "string",
 "VideoMetadata": {
 "Codec": "string",
 "ColorRange": "string",
 "DurationMillis": number,
 "Format": "string",
 "FrameHeight": number,
 "FrameWidth": number,
 "Segment": "string",
 "VideoType": "string"
 }
}
```

```
 "FrameRate": number,
 "FrameWidth": number
 }
}
```

## Elementos de respuesta

Si la acción se realiza correctamente, el servicio devuelve una respuesta HTTP 200.

El servicio devuelve los datos siguientes en formato JSON.

### [JobStatus \(p. 612\)](#)

El estado actual del trabajo de análisis de moderación de contenido.

Type: Cadena

Valores válidos: IN\_PROGRESS | SUCCEEDED | FAILED

### [ModerationLabels \(p. 612\)](#)

Las etiquetas de moderación de contenido detectadas inapropiadas, no deseadas u ofensivas y las horas en que se detectaron.

Type: Matriz de[ContentModerationDetection \(p. 751\)objects](#)

### [ModerationModelVersion \(p. 612\)](#)

Número de versión del modelo de detección de moderación que se utilizó para detectar contenido inapropiado, no deseado u ofensivo.

Type: Cadena

### [NextToken \(p. 612\)](#)

Si la respuesta se trunca, Amazon Rekognition Video devuelve este token que puede utilizar en la solicitud posterior para recuperar el siguiente conjunto de etiquetas de moderación de contenido.

Type: Cadena

Restricciones de longitud: La longitud máxima es de 255 caracteres.

### [StatusMessage \(p. 612\)](#)

Si el trabajo falla, StatusMessage proporciona un mensaje de error descriptivo.

Type: Cadena

### [VideoMetadata \(p. 612\)](#)

Información sobre un vídeo que Amazon Rekognition analizó. Videometadatase devuelve en cada página de respuestas paginadas deGetContentModeration.

Tipo: objeto [VideoMetadata \(p. 845\)](#)

## Errores

### AccessDeniedException

No tiene autorización para realizar la acción.

Código de estado HTTP: 400

#### InternalServerError

Amazon Lex ha tenido un problema de servicio. Pruebe la llamada de nuevo.

Código de estado HTTP: 500

#### InvalidPaginationTokenException

El token de paginación de la solicitud no es válido.

Código de estado HTTP: 400

#### InvalidArgumentException

El parámetro de entrada infringió una restricción. Valide el parámetro antes de llamar a la operación de la API de nuevo.

Código de estado HTTP: 400

#### ProvisionedThroughputExceededException

El número de solicitudes ha superado su límite de rendimiento. Si quieres aumentar este límite, ponte en contacto con Amazon Rekognition.

Código de estado HTTP: 400

#### ResourceNotFoundException

No se encuentra el recurso especificado en la solicitud.

Código de estado HTTP: 400

#### ThrottlingException

Amazon Lex no puede procesar temporalmente la solicitud. Pruebe la llamada de nuevo.

Código de estado HTTP: 500

## Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- [AWS Command Line Interface](#)
- [SDK de AWS para .NET](#)
- [AWS SDK para C++](#)
- [AWS SDK para Go](#)
- [AWSSDK para Java V2](#)
- [AWS SDK para JavaScript](#)
- [SDK de AWS para PHP V3](#)
- [SDK de AWS para Python](#)
- [SDK de AWS para Ruby V3](#)

## GetFaceDetection

Obtiene los resultados de la detección de rostros de un análisis de Amazon Rekognition Video iniciado por [StartFaceDetection](#) (p. 693).

La detección de rostros con Amazon Rekognition Video es una operación asíncrona. Se inicia la detección de rostros llamando [StartFaceDetection](#) (p. 693) que devuelve un identificador de trabajo (`JobId`). Cuando finaliza la operación de detección de rostros, Amazon Rekognition Video publica un estado de finalización en el tema de Amazon Simple Notification Service registrado en la llamada inicial a `StartFaceDetection`. Para obtener los resultados de la operación de detección de rostros, compruebe primero que el valor de estado publicado en el tema de Amazon SNS es `SUCCEEDED`. Si es así, llame [GetFaceDetection](#) (p. 615) y pase el identificador de trabajo (`JobId`) desde la llamada inicial hasta `StartFaceDetection`.

`GetFaceDetection` devuelve una matriz de caras detectadas (`Faces`) ordenados por el momento en que se detectaron las caras.

Utilice el parámetro `MaxResults` para limitar el número de etiquetas devueltas. Si hay más resultados de los especificados en `MaxResults`, el valor de `NextToken` en la respuesta de la operación incluye un token de paginación para obtener el siguiente conjunto de resultados. Para obtener la siguiente página de resultados, llame `GetFaceDetection` y rellénela `NextToken` parámetro de solicitud con el valor del token devuelto desde la llamada anterior a `GetFaceDetection`.

## Sintaxis de la solicitud

```
{
 "JobId": "string",
 "MaxResults": number,
 "NextToken": "string"
}
```

## Parámetros de solicitud

La solicitud acepta los siguientes datos en formato JSON.

### `JobId` (p. 615)

Identificador exclusivo para el trabajo de detección de rostros. La `JobId` devuelto desde `StartFaceDetection`.

Type: Cadena

Restricciones de longitud: Longitud mínima de 1. La longitud máxima es 64.

Patrón: ^[a-zA-Z0-9-\_]+\$

Obligatorio: Sí

### `MaxResults` (p. 615)

Número máximo de resultados que devolver por llamada paginada. El valor más grande que puede especificar es 1000. Si especifica un valor superior a 1 000, se devolverá un máximo de 1 000 resultados. El valor predeterminado es 1000.

Type: Entero

Rango válido: Valor mínimo de 1.

Obligatorio: No

### NextToken (p. 615)

Si la respuesta anterior estaba incompleta (porque hay más caras que recuperar), Amazon Rekognition Video devuelve un token de paginación en la respuesta. Puede utilizar este token de paginación para recuperar el siguiente conjunto de rostros.

Type: Cadena

Restricciones de longitud: La longitud máxima es de 255 caracteres.

Obligatorio: No

## Sintaxis de la respuesta

```
{
 "Faces": [
 {
 "Face": {
 "AgeRange": {
 "High": number,
 "Low": number
 },
 "Beard": {
 "Confidence": number,
 "Value": boolean
 },
 "BoundingBox": {
 "Height": number,
 "Left": number,
 "Top": number,
 "Width": number
 },
 "Confidence": number,
 "Emotions": [
 {
 "Confidence": number,
 "Type": "string"
 }
],
 "Eyeglasses": {
 "Confidence": number,
 "Value": boolean
 },
 "EyesOpen": {
 "Confidence": number,
 "Value": boolean
 },
 "Gender": {
 "Confidence": number,
 "Value": "string"
 },
 "Landmarks": [
 {
 "Type": "string",
 "X": number,
 "Y": number
 }
],
 "MouthOpen": {
 "Confidence": number,
 "Value": boolean
 },
 "Mustache": {
 "Confidence": number,
 "Value": "string"
 }
 }
 }
]
}
```

```
 "Confidence": number,
 "Value": boolean
 },
 "Pose": {
 "Pitch": number,
 "Roll": number,
 "Yaw": number
 },
 "Quality": {
 "Brightness": number,
 "Sharpness": number
 },
 "Smile": {
 "Confidence": number,
 "Value": boolean
 },
 "Sunglasses": {
 "Confidence": number,
 "Value": boolean
 }
},
"Timestamp": number
}
],
"JobStatus": "string",
"NextToken": "string",
"StatusMessage": "string",
"VideoMetadata": {
 "Codec": "string",
 "ColorRange": "string",
 "DurationMillis": number,
 "Format": "string",
 "FrameHeight": number,
 "FrameRate": number,
 "FrameWidth": number
}
}
```

## Elementos de respuesta

Si la acción se realiza correctamente, el servicio devuelve una respuesta HTTP 200.

El servicio devuelve los datos siguientes en formato JSON.

### Faces (p. 616)

Matriz de rostros detectados en el vídeo. Cada elemento incluye los detalles de una cara detectada y el tiempo, en milisegundos desde el principio del vídeo, se detectó el rostro.

Type: Matriz de[FaceDetection](#) (p. 776)objects

### JobStatus (p. 616)

El estado actual del trabajo de detección de rostros.

Type: Cadena

Valores válidos: IN\_PROGRESS | SUCCEEDED | FAILED

### NextToken (p. 616)

Si la respuesta se trunca, Amazon Rekognition devuelve este token que puede utilizar en la solicitud posterior para recuperar el siguiente conjunto de rostros.

Type: Cadena

Restricciones de longitud: La longitud máxima es de 255 caracteres.

[StatusMessage \(p. 616\)](#)

Si el trabajo falla, `StatusMessage` proporciona un mensaje de error descriptivo.

Type: Cadena

[VideoMetadata \(p. 616\)](#)

Información sobre un vídeo que Amazon Rekognition Video analizó. `videometadatase` devuelve en cada página de respuestas paginadas de una operación de vídeo de Amazon Rekognition.

Tipo: objeto [VideoMetadata \(p. 845\)](#)

## Errores

`AccessDeniedException`

No tiene autorización para realizar la acción.

Código de estado HTTP: 400

`InternalServerError`

Amazon Lex ha tenido un problema de servicio. Pruebe la llamada de nuevo.

Código de estado HTTP: 500

`InvalidPaginationTokenException`

El token de paginación de la solicitud no es válido.

Código de estado HTTP: 400

`InvalidParameterException`

El parámetro de entrada infringió una restricción. Valide el parámetro antes de llamar a la operación de la API de nuevo.

Código de estado HTTP: 400

`ProvisionedThroughputExceededException`

El número de solicitudes ha superado su límite de rendimiento. Si quieres aumentar este límite, ponte en contacto con Amazon Rekognition.

Código de estado HTTP: 400

`ResourceNotFoundException`

No se encuentra el recurso especificado en la solicitud.

Código de estado HTTP: 400

`ThrottlingException`

Amazon Lex no puede procesar temporalmente la solicitud. Pruebe la llamada de nuevo.

Código de estado HTTP: 500

## Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- AWS Command Line Interface
- SDK de AWS para .NET
- AWS SDK para C++
- AWS SDK para Go
- AWSSDK para Java V2
- AWS SDK para JavaScript
- SDK de AWS para PHP V3
- SDK de AWS para Python
- SDK de AWS para Ruby V3

## GetFaceSearch

Obtiene los resultados de búsqueda de la búsqueda de rostros para Amazon Rekognition Video iniciada por [StartFaceSearch \(p. 697\)](#). La búsqueda devuelve rostros de una colección que coincidan con los rostros de las personas detectadas en un vídeo. También incluye la (s) hora (s) en que las caras coinciden en el vídeo.

La búsqueda de rostros de en vídeo es una operación asíncrona. Comienza la búsqueda facial llamando a [StartFaceSearch \(p. 697\)](#) que devuelve un identificador de trabajo (JobId). Cuando finaliza la operación de búsqueda, Amazon Rekognition Video publica un estado de finalización en el tema Amazon Simple Notification Service registrado en la llamada inicial `aStartFaceSearch`. Para obtener los resultados de la búsqueda, primero compruebe que el valor de estado publicado en el tema de Amazon SNS es `SUCCEEDED`. Si es así, llame `GetFaceSearch` y pase el identificador de trabajo (JobId) desde la llamada inicial hasta `StartFaceSearch`.

Para obtener más información, consulte [Búsqueda de rostros en una colección \(p. 181\)](#).

Los resultados de la búsqueda se recuperan en una matriz `Persons`, de [PersonMatch \(p. 804\)](#) objetos. Cada `PersonMatch` contiene detalles sobre los rostros que coinciden en la colección de entrada, la información de la persona (atributos faciales, los cuadros delimitadores e identificador de persona) para la persona que coincide,, y el momento en que se encontró una coincidencia de la persona en el vídeo.

### Note

`GetFaceSearch` solo devuelve los atributos faciales predeterminados (`BoundingBox`, `Confidence`, `Landmarks`, `Pose`, `yQuality`). Los demás atributos faciales enumerados en el `Face` se devuelven el objeto de la siguiente sintaxis de respuesta. Para obtener más información, consulte [FaceDetail \(p. 773\)](#).

De forma predeterminada, el `Persons` matriz se ordena por tiempo, en milisegundos desde el principio del vídeo, las personas coinciden. También puede ordenar por personas especificando `INDEX` para la `SORTBY` parámetro de entrada.

## Sintaxis de la solicitud

```
{
 "JobId": "string",
 "MaxResults": number,
 "NextToken": "string",
 "SortBy": "string"
}
```

## Parámetros de solicitud

La solicitud acepta los siguientes datos en formato JSON.

### JobId (p. 620)

Identificador de trabajo de la solicitud de búsqueda. Obtienes el identificador de trabajo de una llamada inicial `aStartFaceSearch`.

Type: Cadena

Restricciones de longitud: Longitud mínima de 1. La longitud máxima es 64.

Patrón: ^[a-zA-Z0-9-\_]+\$

Obligatorio: Sí

### [MaxResults \(p. 620\)](#)

Número máximo de resultados que devolver por llamada paginada. El valor más grande que puede especificar es 1000. Si especifica un valor superior a 1 000, se devolverá un máximo de 1 000 resultados. El valor predeterminado es 1000.

Type: Entero

Rango válido: Valor mínimo de 1.

Obligatorio: No

### [NextToken \(p. 620\)](#)

Si la respuesta anterior estaba incompleta (porque hay más resultados de búsqueda que recuperar), Amazon Rekognition Video devuelve un token de paginación en la respuesta. Puede utilizar este token de paginación para recuperar el siguiente grupo de resultados de búsqueda.

Type: Cadena

Restricciones de longitud: La longitud máxima es de 255 caracteres.

Obligatorio: No

### [SortBy \(p. 620\)](#)

Ordenar para utilizar para agrupar caras en la respuesta. Usar `TIMESTAMP` para agrupar rostros en el momento en que se reconocen. Usar `INDEX` para ordenar por caras reconocidas.

Type: Cadena

Valores válidos: `INDEX` | `TIMESTAMP`

Obligatorio: No

## Sintaxis de la respuesta

```
{
 "JobStatus": "string",
 "NextToken": "string",
 "Persons": [
 {
 "FaceMatches": [
 {
 "Face": {
 "BoundingBox": {
 "Height": number,
 "Left": number,
 "Top": number,
 "Width": number
 },
 "Confidence": number,
 "ExternalImageId": "string",
 "FaceId": "string",
 "ImageId": "string",
 "IndexFacesModelVersion": "string"
 },
 "Similarity": number
 }
],
 "Person": {
 "BoundingBox": {
 "Height": number,
 "Left": number,
 "Top": number,
 "Width": number
 },
 "FaceMatches": [
 {
 "Face": {
 "BoundingBox": {
 "Height": number,
 "Left": number,
 "Top": number,
 "Width": number
 },
 "Confidence": number,
 "ExternalImageId": "string",
 "FaceId": "string",
 "ImageId": "string",
 "IndexFacesModelVersion": "string"
 },
 "Similarity": number
 }
]
 }
 }
]
}
```

```
"Top": number,
"Width": number
},
"Face": {
 "AgeRange": {
 "High": number,
 "Low": number
 },
 "Beard": {
 "Confidence": number,
 "Value": boolean
 },
 "BoundingBox": {
 "Height": number,
 "Left": number,
 "Top": number,
 "Width": number
 },
 "Confidence": number,
 "Emotions": [
 {
 "Confidence": number,
 "Type": "string"
 }
],
 "Eyeglasses": {
 "Confidence": number,
 "Value": boolean
 },
 "EyesOpen": {
 "Confidence": number,
 "Value": boolean
 },
 "Gender": {
 "Confidence": number,
 "Value": "string"
 },
 "Landmarks": [
 {
 "Type": "string",
 "X": number,
 "Y": number
 }
],
 "MouthOpen": {
 "Confidence": number,
 "Value": boolean
 },
 "Mustache": {
 "Confidence": number,
 "Value": boolean
 },
 "Pose": {
 "Pitch": number,
 "Roll": number,
 "Yaw": number
 },
 "Quality": {
 "Brightness": number,
 "Sharpness": number
 },
 "Smile": {
 "Confidence": number,
 "Value": boolean
 },
 "Sunglasses": {
```

```
 "Confidence": number,
 "Value": boolean
 }
},
"Index": number
},
"Timestamp": number
}
],
"StatusMessage": "string",
"VideoMetadata": {
 "Codec": "string",
 "ColorRange": "string",
 "DurationMillis": number,
 "Format": "string",
 "FrameHeight": number,
 "FrameRate": number,
 "FrameWidth": number
}
}
```

## Elementos de respuesta

Si la acción se realiza correctamente, el servicio devuelve una respuesta HTTP 200.

El servicio devuelve los datos siguientes en formato JSON.

### [JobStatus \(p. 621\)](#)

El estado actual del trabajo de búsqueda facial.

Type: Cadena

Valores válidos: IN\_PROGRESS | SUCCEEDED | FAILED

### [NextToken \(p. 621\)](#)

Si la respuesta se trunca, Amazon Rekognition Video devuelve este token que puede utilizar en la solicitud posterior para recuperar el siguiente conjunto de resultados de búsqueda.

Type: Cadena

Restricciones de longitud: La longitud máxima es de 255 caracteres.

### [Persons \(p. 621\)](#)

Una gama de personas, [PersonMatch \(p. 804\)](#), en el vídeo cuyos rostro (s) coinciden con los rostros de una colección de Amazon Rekognition. También incluye información de tiempo para cuándo se emparejan las personas en el vídeo. Especifica la colección de entrada en una llamada inicial `aStartFaceSearch`. Cada `Persons` incluye una hora en la que se ha emparejado la persona, detalles de coincidencia facial (`FaceMatches`) para caras coincidentes de la colección e información de la persona (`Person`) para la persona emparejada.

Type: Matriz de [PersonMatch \(p. 804\)](#) objects

### [StatusMessage \(p. 621\)](#)

Si el trabajo falla, `StatusMessage` proporciona un mensaje de error descriptivo.

Type: Cadena

### [VideoMetadata \(p. 621\)](#)

Información sobre un vídeo que Amazon Rekognition analizó. `VideoMetadata` se devuelve en cada página de respuestas paginadas de una operación de Amazon Rekognition Video.

Tipo: objeto [VideoMetadata \(p. 845\)](#)

## Errores

AccessDeniedException

No tiene autorización para realizar la acción.

Código de estado HTTP: 400

InternalServerError

Amazon Lex ha tenido un problema de servicio. Pruebe la llamada de nuevo.

Código de estado HTTP: 500

InvalidPaginationTokenException

El token de paginación de la solicitud no es válido.

Código de estado HTTP: 400

InvalidParameterException

El parámetro de entrada infringió una restricción. Valide el parámetro antes de llamar a la operación de la API de nuevo.

Código de estado HTTP: 400

ProvisionedThroughputExceededException

El número de solicitudes ha superado su límite de rendimiento. Si quieras aumentar este límite, ponte en contacto con Amazon Rekognition.

Código de estado HTTP: 400

ResourceNotFoundException

No se encuentra el recurso especificado en la solicitud.

Código de estado HTTP: 400

ThrottlingException

Amazon Lex no puede procesar temporalmente la solicitud. Pruebe la llamada de nuevo.

Código de estado HTTP: 500

## Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- [AWS Command Line Interface](#)
- [SDK de AWS para .NET](#)
- [AWS SDK para C++](#)
- [AWS SDK para Go](#)
- [AWSSDK para Java V2](#)
- [AWS SDK para JavaScript](#)
- [SDK de AWS para PHP V3](#)

- [SDK de AWS para Python](#)
- [SDK de AWS para Ruby V3](#)

## GetLabelDetection

Obtiene los resultados de detección de etiquetas de un análisis de Video Rekognition iniciado por [StartLabelDetection \(p. 701\)](#).

La operación de detección de etiquetas se inicia mediante una llamada a [StartLabelDetection \(p. 701\)](#) que devuelve un identificador de trabajo (`JobId`). Cuando finaliza la operación de detección de etiquetas, Amazon Rekognition publica un estado de finalización en el tema de Amazon Simple Notification Service registrado en la llamada inicial `aStartLabelDetection`. Para obtener los resultados de la operación de detección de etiquetas, compruebe primero que el valor de estado publicado en el tema de Amazon SNS es `SUCCEEDED`. Si es así, llame [GetLabelDetection \(p. 626\)](#) y pasa el identificador de trabajo (`JobId`) desde la llamada inicial `aStartLabelDetection`.

`GetLabelDetection` devuelve una matriz de etiquetas detectadas (`Labels`) ordenados por el momento en que se detectaron las etiquetas. También puede ordenar por el nombre de la etiqueta especificando `NAME` para la `SortBy` parámetro de entrada.

Las etiquetas devueltas incluyen el nombre de la etiqueta, el porcentaje de confianza en la precisión de la etiqueta detectada y la hora en que se detectó la etiqueta en el vídeo.

Las etiquetas devueltas también incluyen información de cuadro delimitador para objetos comunes, una taxonomía jerárquica de etiquetas detectadas y la versión del modelo de etiquetas utilizado para la detección.

Utilice el parámetro `MaxResults` para limitar el número de etiquetas devueltas. Si hay más resultados de los especificados en `MaxResults`, el valor de `NextToken` en la respuesta de la operación contiene un token de paginación para obtener el siguiente conjunto de resultados. Para obtener la siguiente página de resultados, llame `GetLabelDetection` y rellénela `NextToken` parámetro `request` con el valor de token devuelto desde la llamada anterior a `GetLabelDetection`.

## Sintaxis de la solicitud

```
{
 "JobId": "string",
 "MaxResults": number,
 "NextToken": "string",
 "SortBy": "string"
}
```

## Parámetros de solicitud

La solicitud acepta los siguientes datos en formato JSON.

### `JobId` (p. 626)

Identificador de Job de la operación de detección de etiquetas para la que desea obtener resultados. Obtienes el identificador de trabajo de una llamada inicial `aStartLabelDetection`.

Type: Cadena

Restricciones de longitud: Longitud mínima de 1. La longitud máxima es 64.

Patrón: ^[a-zA-Z0-9-\_]+\$

Obligatorio: Sí

### `MaxResults` (p. 626)

Número máximo de resultados que devolver por llamada paginada. El valor más grande que puede especificar es 1000. Si especifica un valor superior a 1 000, se devolverá un máximo de 1 000 resultados. El valor predeterminado es 1000.

Type: Entero

Rango válido: Valor mínimo de 1.

Obligatorio: No

[NextToken](#) (p. 626)

Si la respuesta anterior estaba incompleta (porque hay más etiquetas que recuperar), Amazon Rekognition Video devuelve un token de paginación en la respuesta. Puede utilizar este token de paginación para recuperar el siguiente conjunto de etiquetas.

Type: Cadena

Restricciones de longitud: La longitud máxima es de 255 caracteres.

Obligatorio: No

[SortBy](#) (p. 626)

Ordenar para utilizar para los elementos de laLabels matriz. UsarTIMESTAMP para ordenar los elementos de matriz por las etiquetas de tiempo que se detectan. UsarNAME para agrupar alfabéticamente elementos de una etiqueta juntos. Dentro de cada grupo de etiquetas, el elemento de matriz se ordena por confianza de detección. La ordenación predeterminada es portIMESTAMP.

Type: Cadena

Valores válidos: NAME | TIMESTAMP

Obligatorio: No

## Sintaxis de la respuesta

```
{
 "JobStatus": "string",
 "LabelModelVersion": "string",
 "Labels": [
 {
 "Label": {
 "Confidence": number,
 "Instances": [
 {
 "BoundingBox": {
 "Height": number,
 "Left": number,
 "Top": number,
 "Width": number
 },
 "Confidence": number
 }
],
 "Name": "string",
 "Parents": [
 {
 "Name": "string"
 }
]
 },
 "Timestamp": number
 }
],
 "NextToken": "string",
 "StatusMessage": "string",
 "VideoMetadata": {
 ...
 }
}
```

```
 "Codec": "string",
 "ColorRange": "string",
 "DurationMillis": number,
 "Format": "string",
 "FrameHeight": number,
 "FrameRate": number,
 "FrameWidth": number
}
}
```

## Elementos de respuesta

Si la acción se realiza correctamente, el servicio devuelve una respuesta HTTP 200.

El servicio devuelve los datos siguientes en formato JSON.

### [JobStatus \(p. 627\)](#)

El estado actual del trabajo de detección de etiquetas.

Type: Cadena

Valores válidos: IN\_PROGRESS | SUCCEEDED | FAILED

### [LabelModelVersion \(p. 627\)](#)

Número de versión del modelo de detección de etiquetas que se ha utilizado para detectar etiquetas.

Type: Cadena

### [Labels \(p. 627\)](#)

Matriz de etiquetas detectadas en el vídeo. Cada elemento contiene la etiqueta detectada y el tiempo, en milisegundos, transcurrido desde el comienzo del vídeo hasta que se detectó la etiqueta.

Type: Matriz de[LabelDetection \(p. 794\)](#)objects

### [NextToken \(p. 627\)](#)

Si la respuesta se trunca, Amazon Rekognition Video devuelve este token que puede utilizar en la solicitud posterior para recuperar el siguiente conjunto de etiquetas.

Type: Cadena

Restricciones de longitud: La longitud máxima es de 255 caracteres.

### [StatusMessage \(p. 627\)](#)

Si el trabajo falla, StatusMessage proporciona un mensaje de error descriptivo.

Type: Cadena

### [VideoMetadata \(p. 627\)](#)

Información sobre un vídeo que Amazon Rekognition Video analizó. VideoMetadata se devuelve en cada página de respuestas paginadas de una operación de vídeo de Amazon Rekognition.

Tipo: objeto [VideoMetadata \(p. 845\)](#)

## Errores

### AccessDeniedException

No tiene autorización para realizar la acción.

Código de estado HTTP: 400

InternalServerError

Amazon Lex ha tenido un problema de servicio. Pruebe la llamada de nuevo.

Código de estado HTTP: 500

InvalidPaginationTokenException

El token de paginación de la solicitud no es válido.

Código de estado HTTP: 400

InvalidParameterException

El parámetro de entrada infringió una restricción. Valide el parámetro antes de llamar a la operación de la API de nuevo.

Código de estado HTTP: 400

ProvisionedThroughputExceededException

El número de solicitudes ha superado su límite de rendimiento. Si quieres aumentar este límite, ponte en contacto con Amazon Rekognition.

Código de estado HTTP: 400

ResourceNotFoundException

No se encuentra el recurso especificado en la solicitud.

Código de estado HTTP: 400

ThrottlingException

Amazon Lex no puede procesar temporalmente la solicitud. Pruebe la llamada de nuevo.

Código de estado HTTP: 500

## Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- [AWS Command Line Interface](#)
- [SDK de AWS para .NET](#)
- [AWS SDK para C++](#)
- [AWS SDK para Go](#)
- [AWSSDK para Java V2](#)
- [AWS SDK para JavaScript](#)
- [SDK de AWS para PHP V3](#)
- [SDK de AWS para Python](#)
- [SDK de AWS para Ruby V3](#)

## GetPersonTracking

Obtiene los resultados de seguimiento de rutas de un análisis de Amazon Rekognition Video iniciado por [StartPersonTracking \(p. 705\)](#).

La operación de seguimiento de rutas de personas se inicia mediante una llamada `aStartPersonTracking` que devuelve un identificador de trabajo (`JobId`). Cuando finaliza la operación, Amazon Rekognition Video publica un estado de finalización en el tema Amazon Simple Notification Service registrado en la llamada inicial `aStartPersonTracking`.

Para obtener los resultados de la operación de seguimiento de rutas de personas, primero compruebe que el valor de estado publicado en el tema de Amazon SNS es `SUCCEEDED`. Si es así, llame [GetPersonTracking \(p. 630\)](#) y pasa el identificador de trabajo (`JobId`) desde la llamada inicial hasta `aStartPersonTracking`.

`GetPersonTracking` devuelve una matriz `Persons`, de personas rastreadas y la hora (s) de sus trayectorias se rastrearon en el vídeo.

### Note

`GetPersonTracking` solo devuelve los atributos faciales predeterminados (`BoundingBox`, `Confidence`, `Landmarks`, `Pose`, `yQuality`). Los demás atributos faciales enumerados en el `Face` no se devuelven el objeto de la siguiente sintaxis de respuesta. Para obtener más información, consulte [FaceDetail \(p. 773\)](#).

De forma predeterminada, la matriz se ordena por las veces que se realiza el seguimiento de la ruta de una persona en el vídeo. Puede ordenar por personas con seguimiento especificando `INDEX` para la `SortBy` parámetro de entrada.

Usar `MaxResults` para limitar el número de elementos devueltos. Si hay más resultados de los especificados en `MaxResults`, el valor de `NextToken` en la respuesta de la operación contiene un token de paginación para obtener el siguiente conjunto de resultados. Para obtener la siguiente página de resultados, llame `GetPersonTracking` y rellénela `NextToken` El parámetro `request` con el valor del token devuelto en la llamada anterior a `GetPersonTracking`.

## Sintaxis de la solicitud

```
{
 "JobId": "string",
 "MaxResults": number,
 "NextToken": "string",
 "SortBy": "string"
}
```

## Parámetros de solicitud

La solicitud acepta los siguientes datos en formato JSON.

### `JobId` (p. 630)

El identificador de un trabajo que rastrea a personas en un vídeo. Obtiene el `JobId` de una llamada `aStartPersonTracking`.

Type: Cadena

Restricciones de longitud: Longitud mínima de 1. La longitud máxima es 64.

Patrón: ^[a-zA-Z0-9-\_]+\$

Obligatorio: Sí

#### [MaxResults \(p. 630\)](#)

Número máximo de resultados que devolver por llamada paginada. El valor más grande que puede especificar es 1000. Si especifica un valor superior a 1 000, se devolverá un máximo de 1 000 resultados. El valor predeterminado es 1000.

Type: Entero

Rango válido: Valor mínimo de 1.

Obligatorio: No

#### [NextToken \(p. 630\)](#)

Si la respuesta anterior estaba incompleta (porque hay más personas que recuperar), Amazon Rekognition Video devuelve un token de paginación en la respuesta. Puede utilizar este token de paginación para recuperar el siguiente conjunto de personas.

Type: Cadena

Restricciones de longitud: La longitud máxima es de 255 caracteres.

Obligatorio: No

#### [SortBy \(p. 630\)](#)

Ordenar para utilizar para los elementos de laPersons matriz. UsarTIMESTAMP para ordenar los elementos de matriz por el momento en que se detectan personas. UsarINDEX para ordenar por las personas rastreadas. Si ordenas porINDEX, los elementos de matriz de cada persona se ordenan por confianza de detección. La ordenación predeterminada es porTIMESTAMP.

Type: Cadena

Valores válidos: INDEX | TIMESTAMP

Obligatorio: No

## Sintaxis de la respuesta

```
{
 "JobStatus": "string",
 "NextToken": "string",
 "Persons": [
 {
 "Person": {
 "BoundingBox": {
 "Height": number,
 "Left": number,
 "Top": number,
 "Width": number
 },
 "Face": {
 "AgeRange": {
 "High": number,
 "Low": number
 },
 "Beard": {
 "Confidence": number,
 "Value": boolean
 },
 "BoundingBox": {
 "Height": number,
 "Left": number,
 "Top": number,
 "Width": number
 }
 }
 }
 }
]
}
```

```
 "Height": number,
 "Left": number,
 "Top": number,
 "Width": number
 },
 "Confidence": number,
 "Emotions": [
 {
 "Confidence": number,
 "Type": "string"
 }
],
 "Eyeglasses": {
 "Confidence": number,
 "Value": boolean
 },
 "EyesOpen": {
 "Confidence": number,
 "Value": boolean
 },
 "Gender": {
 "Confidence": number,
 "Value": "string"
 },
 "Landmarks": [
 {
 "Type": "string",
 "X": number,
 "Y": number
 }
],
 "MouthOpen": {
 "Confidence": number,
 "Value": boolean
 },
 "Mustache": {
 "Confidence": number,
 "Value": boolean
 },
 "Pose": {
 "Pitch": number,
 "Roll": number,
 "Yaw": number
 },
 "Quality": {
 "Brightness": number,
 "Sharpness": number
 },
 "Smile": {
 "Confidence": number,
 "Value": boolean
 },
 "Sunglasses": {
 "Confidence": number,
 "Value": boolean
 }
},
 "Index": number
},
 "Timestamp": number
}
],
 "StatusMessage": "string",
 "VideoMetadata": {
 "Codec": "string",
 "ColorRange": "string",
 "FrameRate": number
}
}
```

```
 "DurationMillis": number,
 "Format": "string",
 "FrameHeight": number,
 "FrameRate": number,
 "FrameWidth": number
}
}
```

## Elementos de respuesta

Si la acción se realiza correctamente, el servicio devuelve una respuesta HTTP 200.

El servicio devuelve los datos siguientes en formato JSON.

### [JobStatus \(p. 631\)](#)

El estado actual del trabajo de seguimiento de personas.

Type: Cadena

Valores válidos: IN\_PROGRESS | SUCCEEDED | FAILED

### [NextToken \(p. 631\)](#)

Si la respuesta se trunca, Amazon Rekognition Video devuelve este token que puede utilizar en la solicitud posterior para recuperar el siguiente grupo de personas.

Type: Cadena

Restricciones de longitud: La longitud máxima es de 255 caracteres.

### [Persons \(p. 631\)](#)

Se realizó un seguimiento de una matriz de personas detectadas en el vídeo y el momento en que se realizó el seguimiento de su ruta a lo largo del vídeo. Un elemento de matriz existirá cada vez que se realiza el seguimiento de la ruta de una persona.

Type: Matriz de[PersonDetection \(p. 803\)](#)objects

### [StatusMessage \(p. 631\)](#)

Si el trabajo falla, StatusMessage proporciona un mensaje de error descriptivo.

Type: Cadena

### [VideoMetadata \(p. 631\)](#)

Información sobre un vídeo que Amazon Rekognition Video analizó. VideoMetadata se devuelve en cada página de respuestas paginadas de una operación de Amazon Rekognition Video.

Tipo: objeto [VideoMetadata \(p. 845\)](#)

## Errores

### AccessDeniedException

No tiene autorización para realizar la acción.

Código de estado HTTP: 400

### InternalServerError

Amazon Lex ha tenido un problema de servicio. Pruebe la llamada de nuevo.

Código de estado HTTP: 500

InvalidPaginationTokenException

El token de paginación de la solicitud no es válido.

Código de estado HTTP: 400

InvalidParameterException

El parámetro de entrada infringió una restricción. Valide el parámetro antes de llamar a la operación de la API de nuevo.

Código de estado HTTP: 400

ProvisionedThroughputExceededException

El número de solicitudes ha superado su límite de rendimiento. Si necesita aumentar este límite, póngase en contacto con Amazon Rekognition.

Código de estado HTTP: 400

ResourceNotFoundException

No se encuentra el recurso especificado en la solicitud.

Código de estado HTTP: 400

ThrottlingException

Amazon Lex no puede procesar temporalmente la solicitud. Pruebe la llamada de nuevo.

Código de estado HTTP: 500

## Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- [AWS Command Line Interface](#)
- [SDK de AWS para .NET](#)
- [AWS SDK para C++](#)
- [AWS SDK para Go](#)
- [AWSSDK para Java V2](#)
- [AWS SDK para JavaScript](#)
- [SDK de AWS para PHP V3](#)
- [SDK de AWS para Python](#)
- [SDK de AWS para Ruby V3](#)

## GetSegmentDetection

Obtiene los resultados de la detección de segmentos de un análisis de Video Rekognition iniciado por [StartSegmentDetection \(p. 712\)](#).

La detección de segmentos con Amazon Rekognition Video es una operación asíncrona. Se inicia la detección de segmentos llamando [StartSegmentDetection \(p. 712\)](#) que devuelve un identificador de trabajo (`JobId`). Cuando finaliza la operación de detección de segmentos, Amazon Rekognition publica un estado de finalización en el tema de Amazon Simple Notification Service registrado en la llamada inicial `aStartSegmentDetection`. Para obtener los resultados de la operación de detección de segmentos, compruebe primero que el valor de estado publicado en el tema de Amazon SNS es `SUCCEEDED`. si es así, llame [GetSegmentDetection](#) y pase el identificador de trabajo (`JobId`) de la convocatoria inicial `deStartSegmentDetection`.

`GetSegmentDetection` devuelve los segmentos detectados en una matriz (`Segments`) de [SegmentDetection \(p. 818\)](#) objetos. `Segments` se ordena por los tipos de segmentos especificados en el `SegmentTypes` parámetro de entrada `deStartSegmentDetection`. Cada elemento de la matriz incluye el segmento detectado, la confianza del porcentaje en la precisión del segmento detectado, el tipo de segmento y el fotograma en el que se detectó el segmento.

Usar `SelectedSegmentTypes` para averiguar el tipo de detección de segmentos solicitada en la llamada `aStartSegmentDetection`.

Usar `MaxResults` para limitar el número de detecciones de segmentos devueltos. Si hay más resultados de los especificados en `MaxResults`, el valor de `NextToken` en la respuesta de la operación contiene un token de paginación para obtener el siguiente conjunto de resultados. Para obtener la siguiente página de resultados, llame `GetSegmentDetection` y rellénela `NextToken` con el valor de token devuelto en la llamada anterior `GetSegmentDetection`.

Para obtener más información, consulte [Detección de segmentos de vídeo en vídeo almacenado \(p. 338\)](#).

## Sintaxis de la solicitud

```
{
 "JobId": "string",
 "MaxResults": number,
 "NextToken": "string"
}
```

## Parámetros de solicitud

La solicitud acepta los siguientes datos en formato JSON.

### `JobId` (p. 635)

Identificador de Job de la operación de detección de texto para la que desea obtener resultados. Obtienes el identificador de trabajo de una llamada inicial `aStartSegmentDetection`.

Type: Cadena

Restricciones de longitud: Longitud mínima de 1. La longitud máxima es 64.

Patrón: ^[a-zA-Z0-9-\_]+\$

Obligatorio: Sí

### `MaxResults` (p. 635)

Número máximo de resultados que devolver por llamada paginada. El valor más grande que puede especificar es 1000.

Type: Entero

Rango válido: Valor mínimo de 1.

Obligatorio: No

[NextToken \(p. 635\)](#)

Si la respuesta se trunca, Amazon Rekognition Video devuelve este token que puede utilizar en la solicitud posterior para recuperar el siguiente conjunto de texto.

Type: Cadena

Restricciones de longitud: La longitud máxima es de 255 caracteres.

Obligatorio: No

## Sintaxis de la respuesta

```
{
 "AudioMetadata": [
 {
 "Codec": "string",
 "DurationMillis": number,
 "NumberOfChannels": number,
 "SampleRate": number
 }
],
 "JobStatus": "string",
 "NextToken": "string",
 "Segments": [
 {
 "DurationFrames": number,
 "DurationMillis": number,
 "DurationSMPTE": "string",
 "EndFrameNumber": number,
 "EndTimecodeSMPTE": "string",
 "EndTimestampMillis": number,
 "ShotSegment": {
 "Confidence": number,
 "Index": number
 },
 "StartFrameNumber": number,
 "StartTimecodeSMPTE": "string",
 "StartTimestampMillis": number,
 "TechnicalCueSegment": {
 "Confidence": number,
 "Type": "string"
 },
 "Type": "string"
 }
],
 "SelectedSegmentTypes": [
 {
 "ModelVersion": "string",
 "Type": "string"
 }
],
 "StatusMessage": "string",
 "VideoMetadata": [
 {
 "Codec": "string",
 "ColorRange": "string",
 "Width": number,
 "Height": number
 }
]
}
```

```
 "DurationMillis": number,
 "Format": "string",
 "FrameHeight": number,
 "FrameRate": number,
 "FrameWidth": number
 }
}
```

## Elementos de respuesta

Si la acción se realiza correctamente, el servicio devuelve una respuesta HTTP 200.

El servicio devuelve los datos siguientes en formato JSON.

### [AudioMetadata \(p. 636\)](#)

Una matriz de objetos [AudioMetadata \(p. 737\)](#). 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`.

Type: Matriz de[AudioMetadata \(p. 737\)objects](#)

### [JobStatus \(p. 636\)](#)

Estado actual del trabajo de detección de segmentos.

Type: Cadena

Valores válidos: `IN_PROGRESS` | `SUCCEEDED` | `FAILED`

### [NextToken \(p. 636\)](#)

Si la respuesta anterior estaba incompleta (porque hay más etiquetas que recuperar), Amazon Rekognition Video devuelve un token de paginación en la respuesta. Puede utilizar este token de paginación para recuperar el siguiente conjunto de texto.

Type: Cadena

Restricciones de longitud: La longitud máxima es de 255 caracteres.

### [Segments \(p. 636\)](#)

Matriz de segmentos detectados en un vídeo. 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.

Type: Matriz de[SegmentDetection \(p. 818\)objects](#)

### [SelectedSegmentTypes \(p. 636\)](#)

Matriz que contiene los tipos de segmentos solicitados en la llamada `aStartSegmentDetection`.

Type: Matriz de[SegmentTypeInfo \(p. 821\)objects](#)

### [StatusMessage \(p. 636\)](#)

Si el trabajo falla, `StatusMessage` proporciona un mensaje de error descriptivo.

Type: Cadena

### [VideoMetadata \(p. 636\)](#)

En la actualidad, Amazon Rekognition Video devuelve un solo[VideoMetadata \(p. 845\)](#)un objeto de videoMetadata matriz. 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.

Type: Matriz de[VideoMetadata \(p. 845\)](#)objects

## Errores

### AccessDeniedException

No tiene autorización para realizar la acción.

Código de estado HTTP: 400

### InternalServerError

Amazon Lex ha tenido un problema de servicio. Pruebe la llamada de nuevo.

Código de estado HTTP: 500

### InvalidPaginationTokenException

El token de paginación de la solicitud no es válido.

Código de estado HTTP: 400

### InvalidParameterException

El parámetro de entrada infringió una restricción. Valide el parámetro antes de llamar a la operación de la API de nuevo.

Código de estado HTTP: 400

### ProvisionedThroughputExceededException

El número de solicitudes ha superado su límite de rendimiento. Si quieres aumentar este límite, ponte en contacto con Amazon Rekognition.

Código de estado HTTP: 400

### ResourceNotFoundException

No se encuentra el recurso especificado en la solicitud.

Código de estado HTTP: 400

### ThrottlingException

Amazon Lex no puede procesar temporalmente la solicitud. Pruebe la llamada de nuevo.

Código de estado HTTP: 500

## Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- [AWS Command Line Interface](#)

- [SDK de AWS para .NET](#)
- [AWS SDK para C++](#)
- [AWS SDK para Go](#)
- [AWSSDK para Java V2](#)
- [AWS SDK para JavaScript](#)
- [SDK de AWS para PHP V3](#)
- [SDK de AWS para Python](#)
- [SDK de AWS para Ruby V3](#)

## GetTextDetection

Obtiene los resultados de detección de texto de un análisis Amazon Rekognition Video iniciado por [StartTextDetection](#) (p. 718).

La detección de texto con Amazon Rekognition Video es una operación asíncrona. Para iniciar la detección de texto llamando [StartTextDetection](#) (p. 718) que devuelve un identificador de trabajo (`JobId`) Cuando finaliza la operación de detección de texto, Amazon Rekognition publica un estado de finalización en el tema de Amazon Simple Notification Service registrado en la llamada inicial a `aStartTextDetection`. Para obtener los resultados de la operación de detección de texto, compruebe primero que el valor de estado publicado en el tema de Amazon SNS es `SUCCEEDED`. Si es así, llame `GetTextDetection` y pase el identificador de trabajo (`JobId`) de la convocatoria inicial `aStartLabelDetection`.

`GetTextDetection` devuelve una matriz de texto detectado (`TextDetections`) ordenados por el momento en que se detectó el texto, hasta 50 palabras por fotograma de vídeo.

Cada elemento de la matriz incluye el texto detectado, el porcentaje de confianza en la precisión del texto detectado, la hora en que se detectó el texto, la información del cuadro delimitador para la ubicación del texto e identificadores únicos para las palabras y sus líneas.

Utilice el parámetro `MaxResults` para limitar el número de detecciones de texto devueltas. Si hay más resultados de los especificados en `MaxResults`, el valor de `NextToken` en la respuesta de la operación contiene un token de paginación para obtener el siguiente conjunto de resultados. Para obtener la siguiente página de resultados, llame `GetTextDetection` y rellénela `NextToken` parámetro de solicitud con el valor de token devuelto de la llamada anterior a `GetTextDetection`.

## Sintaxis de la solicitud

```
{
 "JobId": "string",
 "MaxResults": number,
 "NextToken": "string"
}
```

## Parámetros de solicitud

La solicitud acepta los siguientes datos en formato JSON.

### `JobId` (p. 640)

Identificador de Job de la operación de detección de texto para la que desea obtener resultados. Obtienes el identificador de trabajo de una llamada inicial `aStartTextDetection`.

Type: Cadena

Restricciones de longitud: Longitud mínima de 1. La longitud máxima es 64.

Patrón: ^[a-zA-Z0-9-\_]+\$

Obligatorio: Sí

### `MaxResults` (p. 640)

Número máximo de resultados que devolver por llamada paginada. El valor más grande que puede especificar es 1000.

Type: Entero

Rango válido: Valor mínimo de 1.

Obligatorio: No

[NextToken](#) (p. 640)

Si la respuesta anterior estaba incompleta (porque hay más etiquetas que recuperar), Amazon Rekognition Video devuelve un token de paginación en la respuesta. Puede utilizar este token de paginación para recuperar el siguiente conjunto de texto.

Type: Cadena

Restricciones de longitud: La longitud máxima es de 255 caracteres.

Obligatorio: No

## Sintaxis de la respuesta

```
{
 "JobStatus": "string",
 "NextToken": "string",
 "StatusMessage": "string",
 "TextDetections": [
 {
 "TextDetection": {
 "Confidence": number,
 "DetectedText": "string",
 "Geometry": {
 "BoundingBox": {
 "Height": number,
 "Left": number,
 "Top": number,
 "Width": number
 },
 "Polygon": [
 {
 "x": number,
 "y": number
 }
]
 },
 "Id": number,
 "ParentId": number,
 "Type": "string"
 },
 "Timestamp": number
 }
],
 "TextModelVersion": "string",
 "VideoMetadata": {
 "Codec": "string",
 "ColorRange": "string",
 "DurationMillis": number,
 "Format": "string",
 "FrameHeight": number,
 "FrameRate": number,
 "FrameWidth": number
 }
}
```

## Elementos de respuesta

Si la acción se realiza correctamente, el servicio devuelve una respuesta HTTP 200.

El servicio devuelve los datos siguientes en formato JSON.

[JobStatus \(p. 641\)](#)

Estado actual del trabajo de detección de texto.

Type: Cadena

Valores válidos: IN\_PROGRESS | SUCCEEDED | FAILED

[NextToken \(p. 641\)](#)

Si la respuesta se trunca, Amazon Rekognition Video devuelve este token que puede utilizar en la solicitud posterior para recuperar el siguiente conjunto de texto.

Type: Cadena

Restricciones de longitud: La longitud máxima es de 255 caracteres.

[StatusMessage \(p. 641\)](#)

Si el trabajo falla, StatusMessage proporciona un mensaje de error descriptivo.

Type: Cadena

[TextDetections \(p. 641\)](#)

Matriz de texto detectada en el vídeo. Cada elemento contiene el texto detectado, el tiempo en milisegundos desde el inicio del vídeo en que se detectó el texto y dónde se detectó en la pantalla.

Type: Matriz de [TextDetectionResult \(p. 839\)](#) objects

[TextModelVersion \(p. 641\)](#)

Número de versión del modelo de detección de texto que se utilizó para detectar texto.

Type: Cadena

[VideoMetadata \(p. 641\)](#)

Información sobre un vídeo que Amazon Rekognition analizó. VideoMetadata se devuelve en cada página de respuestas paginadas de una operación de vídeo de Amazon Rekognition.

Tipo: objeto [VideoMetadata \(p. 845\)](#)

## Errores

AccessDeniedException

No tiene autorización para realizar la acción.

Código de estado HTTP: 400

InternalServerError

Amazon Lex ha tenido un problema de servicio. Pruebe la llamada de nuevo.

Código de estado HTTP: 500

InvalidPaginationTokenException

El token de paginación de la solicitud no es válido.

Código de estado HTTP: 400

#### InvalidArgumentException

El parámetro de entrada infringió una restricción. Valide el parámetro antes de llamar a la operación de la API de nuevo.

Código de estado HTTP: 400

#### ProvisionedThroughputExceededException

El número de solicitudes ha superado su límite de rendimiento. Si quieres aumentar este límite, ponte en contacto con Amazon Rekognition.

Código de estado HTTP: 400

#### ResourceNotFoundException

No se encuentra el recurso especificado en la solicitud.

Código de estado HTTP: 400

#### ThrottlingException

Amazon Lex no puede procesar temporalmente la solicitud. Pruebe la llamada de nuevo.

Código de estado HTTP: 500

## Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- [AWS Command Line Interface](#)
- [SDK de AWS para .NET](#)
- [AWS SDK para C++](#)
- [AWS SDK para Go](#)
- [AWSSDK para Java V2](#)
- [AWS SDK para JavaScript](#)
- [SDK de AWS para PHP V3](#)
- [SDK de AWS para Python](#)
- [SDK de AWS para Ruby V3](#)

## IndexFaces

Detecta rostros en la imagen de entrada y los añade a la colección especificada.

Amazon Rekognition no guarda las caras reales detectadas. En lugar de ello, el algoritmo de detección subyacente detecta primero los rostros en la imagen de entrada. Para cada rostro, el algoritmo extrae los rasgos faciales en un vector de rasgos de rasgos y los almacena en la base de datos backend. Amazon Rekognition utiliza vectores de entidades cuando realiza operaciones de búsqueda y coincidencia de rostros mediante el[SearchFaces \(p. 676\)](#)y[SearchFacesByImage \(p. 680\)](#)operaciones.

Para obtener más información, consulte [Incorporación de rostros en una colección \(p. 204\)](#).

Para obtener el número de rostros de una colección, llame a[DescribeCollection \(p. 556\)](#).

Si utilizas la versión 1.0 del modelo de detección de rostros, `IndexFaces`indexa las 15 caras más grandes de la imagen de entrada. Las versiones posteriores del modelo de detección de rostros indexan los 100 rostros de mayor tamaño de la imagen de entrada.

Si utilizas la versión 4 o posterior del modelo de cara, la información de orientación de la imagen no se devuelve en el[OrientationCorrection](#).

Para determinar qué versión del modelo está utilizando, llame a[DescribeCollection \(p. 556\)](#)y proporciona el ID de colección. También puede obtener la versión del modelo desde el valor de[FaceModelVersion](#)en la respuesta de[IndexFaces](#)

Para obtener más información, consulte [Control de versiones del modelo \(p. 10\)](#).

Si proporciona la opción opcional[ExternalImageId](#)para la imagen de entrada que proporcionaste, Amazon Rekognition asocia este ID con todas las caras que detecta. Cuando llamas al[ListFaces \(p. 663\)](#), la respuesta devuelve el ID externo. Puede utilizar este ID de imagen externo para crear un índice del lado del cliente para asociar las caras a cada imagen. A continuación, puede utilizar el índice para buscar todas las caras de una imagen.

Puede especificar el número máximo de rostros que se indexará con la[MaxFaces](#)parámetro de entrada. Esto resulta útil cuando se desea indexar los rostros de mayor tamaño de una imagen, pero no los más pequeños, como, por ejemplo, los que pertenecen a las personas que están de pie en segundo plano.

La[QualityFilter](#)el parámetro de entrada le permite filtrar las caras detectadas que no cumplen con la barra de calidad requerida. La barra de calidad se basa en una variedad de casos de uso comunes. De forma predeterminada, `IndexFaces`selecciona la barra de calidad que se utiliza para filtrar los rostros. También puede elegir explícitamente la barra de calidad. Usar[QualityFilter](#), para configurar la barra de calidad especificandolo[LOW](#),[MEDIUM](#), o bien[HIGH](#). Si no desea filtrar las caras detectadas, especifiquen[NONE](#).

### Note

Para utilizar el filtrado según la calidad, necesita una colección asociada a la versión 3 del modelo de rostros o posterior. Para obtener la versión del modelo de rostros asociado a una colección, llame a [DescribeCollection \(p. 556\)](#).

La información sobre los rostros detectados en una imagen, pero no indexada, se devuelve en una matriz de[UnindexedFace \(p. 842\)](#)objetos,[UnindexedFaces](#). Las caras no se indexan por razones tales como:

- El número de rostros detectados supera el valor de[MaxFaces](#)parámetro de solicitud.
- 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.

En respuesta, el `IndexFaces` devuelve una matriz de metadatos para todas las caras detectadas, `FaceRecords`. Esto incluye:

- El cuadro delimitador, `BoundingBox`, de la cara detectada.
- Un valor de confianza, `Confidence`, que indica la confianza de que el cuadro delimitador contiene una cara.
- Una identificación facial, `FaceId`, asignado por el servicio para cada rostro detectado y almacenado.
- Un ID de imagen, `ImageId`, asignado por el servicio para la imagen de entrada.

Si solicita todos los atributos faciales (utilizando el `detectionAttributes` parámetro), Amazon Rekognition devuelve atributos faciales detallados, como hitos faciales (por ejemplo, ubicación del ojo y la boca) y otros atributos faciales. Si proporciona la misma imagen, especifique la misma colección y utilice el mismo ID externo en el `IndexFaces`, Amazon Rekognition no guarda metadatos de cara duplicados.

La imagen de entrada se pasa como bytes de imagen codificados en base64 o como referencia a una imagen de un bucket de Amazon S3. Si utiliza la CLI de AWS para llamar a las operaciones de Amazon Rekognition, no es posible transferir bytes de imágenes. La imagen debe estar formateada como archivo PNG o JPEG.

Esta operación requiere permisos para realizar la acción `rekognition:IndexFaces`.

## Sintaxis de la solicitud

```
{
 "CollectionId": "string",
 "DetectionAttributes": ["string"],
 "ExternalImageId": "string",
 "Image": {
 "Bytes": blob,
 "S3Object": {
 "Bucket": "string",
 "Name": "string",
 "Version": "string"
 }
 },
 "MaxFaces": number,
 "QualityFilter": "string"
}
```

## Parámetros de solicitud

La solicitud acepta los siguientes datos en formato JSON.

### `CollectionId` (p. 645)

Id. de una colección existente a la que desea agregar las caras detectadas en las imágenes de entrada.

Type: Cadena

Restricciones de longitud: Longitud mínima de 1. La longitud máxima es de 255 caracteres.

Patrón: [a-zA-Z0-9\_.\-\-]+

Obligatorio: Sí

[DetectionAttributes \(p. 645\)](#)

Una serie de atributos faciales que desea que se devuelvan. Puede ser la lista predeterminada de atributos o todos los atributos. Si no especifica un valor para `Attributes` o si especifica [ "DEFAULT" ], la API devuelve el siguiente subconjunto de atributos faciales: `BoundingBox`, `Confidence`, `Pose`, `Quality`, y `Landmarks`. Si proporciona [ "ALL" ], se devuelven todos los atributos faciales, pero la operación tarda más en completarse.

Si proporciona ambas cosas, [ "ALL", "DEFAULT" ], el servicio utiliza un operador AND lógico para determinar qué atributos devolver (en este caso, todos los atributos).

Type: Matriz de cadenas

Valores válidos: `DEFAULT` | `ALL`

Obligatorio: No

[ExternalImageId \(p. 645\)](#)

ID que desea asignar a todas las caras detectadas en la imagen.

Type: Cadena

Restricciones de longitud: Longitud mínima de 1. La longitud máxima es de 255 caracteres.

Patrón: [ a-zA-Z0-9\_.\-\:\ ]+

Obligatorio: No

[Image \(p. 645\)](#)

La imagen de entrada como bytes codificados en base64 o un objeto S3. Si utiliza la CLI de AWS para llamar a las operaciones de Amazon Rekognition, no se admite la transferencia de bytes de imagen codificados en base64.

Si utiliza un SDK de AWS para llamar a Amazon Rekognition, es posible que no tenga que codificar en base 64 bytes de imagen pasados mediante `Bytes`. Para obtener más información, consulte [Especificaciones de imágenes \(p. 29\)](#).

Tipo: objeto [Image \(p. 786\)](#)

Obligatorio: Sí

[MaxFaces \(p. 645\)](#)

El número máximo de rostros que se debe indexar. El valor de `MaxFaces` debe ser mayor o igual que 1. `IndexFaces` devuelve no más de 100 caras detectadas en una imagen, incluso si especifica un valor mayor para `MaxFaces`.

Si `IndexFaces` detecta más caras que el valor de `MaxFaces`, las caras con la menor calidad se filtran primero. Si aún hay más caras que el valor de `MaxFaces`, las caras con los cuadros delimitadores más pequeños se filtran (hasta el número necesario para satisfacer el valor de `MaxFaces`). La información sobre las caras no indexadas está disponible en el `UnindexedFaces` matriz.

Las caras devueltas por `IndexFaces` se ordenan por el tamaño de cuadro delimitador de cara más grande al tamaño más pequeño, en orden descendente.

`MaxFaces` se puede utilizar con una colección asociada a cualquier versión del modelo de rostros.

Type: Entero

Rango válido: Valor mínimo de 1.

Obligatorio: No

### QualityFilter (p. 645)

Filtro que especifica una barra de calidad para determinar cuánto filtrado se realiza para identificar caras. Las caras filtradas no están indexadas. Si especifica AUTO, Amazon Rekognition elige la barra de calidad. Si especifica LOW, MEDIUM, o bien HIGH, el filtrado elimina todas las caras que no cumplen la barra de calidad elegida. El valor predeterminado es AUTO. La barra de calidad se basa en una variedad de casos de uso comunes. Las detecciones de baja calidad pueden ocurrir por diversas razones. Algunos ejemplos son un objeto que se identifica mal como cara, cara demasiado borrosa o cara con una pose demasiado extrema para usarlo. Si especifica NONE, no se realiza ningún filtrado.

Para utilizar el filtrado según la calidad, la colección que está utilizando debe asociarse a la versión 3 del modelo de rostros o posterior.

Type: Cadena

Valores válidos: NONE | AUTO | LOW | MEDIUM | HIGH

Obligatorio: No

## Sintaxis de la respuesta

```
{
 "FaceModelVersion": "string",
 "FaceRecords": [
 {
 "Face": {
 "BoundingBox": {
 "Height": number,
 "Left": number,
 "Top": number,
 "Width": number
 },
 "Confidence": number,
 "ExternalImageId": "string",
 "FaceId": "string",
 "ImageId": "string",
 "IndexFacesModelVersion": "string"
 },
 "FaceDetail": {
 "AgeRange": {
 "High": number,
 "Low": number
 },
 "Beard": {
 "Confidence": number,
 "Value": boolean
 },
 "BoundingBox": {
 "Height": number,
 "Left": number,
 "Top": number,
 "Width": number
 },
 "Confidence": number,
 "Emotions": [
 {
 "Confidence": number,
 "Type": "string"
 }
],
 "Eyeglasses": {
 "Confidence": number,
 "Type": "string"
 }
 }
 }
]
}
```

```
 "Value": boolean
 },
 "EyesOpen": {
 "Confidence": number,
 "Value": boolean
 },
 "Gender": {
 "Confidence": number,
 "Value": string
 },
 "Landmarks": [
 {
 "Type": string,
 "X": number,
 "Y": number
 }
],
 "MouthOpen": {
 "Confidence": number,
 "Value": boolean
 },
 "Mustache": {
 "Confidence": number,
 "Value": boolean
 },
 "Pose": {
 "Pitch": number,
 "Roll": number,
 "Yaw": number
 },
 "Quality": {
 "Brightness": number,
 "Sharpness": number
 },
 "Smile": {
 "Confidence": number,
 "Value": boolean
 },
 "Sunglasses": {
 "Confidence": number,
 "Value": boolean
 }
}
],
"OrientationCorrection": string,
"UnindexedFaces": [
 {
 "FaceDetail": {
 "AgeRange": {
 "High": number,
 "Low": number
 },
 "Beard": {
 "Confidence": number,
 "Value": boolean
 },
 "BoundingBox": {
 "Height": number,
 "Left": number,
 "Top": number,
 "Width": number
 },
 "Confidence": number,
 "Emotions": [
 {

```

```
 "Confidence": number,
 "Type": "string"
 }
],
"Eyeglasses": {
 "Confidence": number,
 "Value": boolean
},
"EyesOpen": {
 "Confidence": number,
 "Value": boolean
},
"Gender": {
 "Confidence": number,
 "Value": "string"
},
"Landmarks": [
 {
 "Type": "string",
 "X": number,
 "Y": number
 }
],
"MouthOpen": {
 "Confidence": number,
 "Value": boolean
},
"Mustache": {
 "Confidence": number,
 "Value": boolean
},
"Pose": {
 "Pitch": number,
 "Roll": number,
 "Yaw": number
},
"Quality": {
 "Brightness": number,
 "Sharpness": number
},
"Smile": {
 "Confidence": number,
 "Value": boolean
},
"Sunglasses": {
 "Confidence": number,
 "Value": boolean
}
],
"Reasons": ["string"]
}
]
```

## Elementos de respuesta

Si la acción se realiza correctamente, el servicio devuelve una respuesta HTTP 200.

El servicio devuelve los datos siguientes en formato JSON.

### FaceModelVersion (p. 647)

El número de versión del modelo de detección de rostros asociado a la colección de entrada (`CollectionId`).

Type: Cadena

[FaceRecords \(p. 647\)](#)

Matriz de rostros detectados y añadidos a la colección. Para obtener más información, consulte [Administración de rostros en una colección \(p. 182\)](#).

Type: Matriz de[FaceRecord \(p. 778\)](#)objects

[OrientationCorrection \(p. 647\)](#)

Si la colección está asociada a un modelo de detección de rostros posterior a la versión 3.0, el valor de[OrientationCorrection](#)siempre es nulo y no se devuelve información de orientación.

Si la colección está asociada a un modelo de detección de rostros de la versión 3.0 o anterior, se aplica lo siguiente:

- Si la imagen de entrada está en formato.jpeg, podría contener metadatos de formato de archivo de imagen intercambiable (EXIF) intercambiables que incluyen la orientación de la imagen. Amazon Rekognition utiliza esta información de orientación para realizar la corrección de imagen: las coordenadas del cuadro delimitador se traducen para representar las ubicaciones de los objetos después de utilizar la información de orientación de los metadatos Exif para corregir la orientación de la imagen. Las imágenes en formato.png no contienen metadatos Exif. El valor de[OrientationCorrection](#)nulo.
- Si la imagen no contiene información de orientación en sus metadatos Exif, Amazon Rekognition devuelve una orientación estimada (ROTATE\_0, ROTATE\_90, ROTATE\_180, ROTATE\_270). Amazon Rekognition no realiza la corrección de imagen de las imágenes. Las coordenadas del cuadro delimitador no se traducen y representan las ubicaciones de los objetos antes de girar la imagen.

La información del cuadro delimitador se devuelve en el[FaceRecords](#)matriz. Puede obtener la versión del modelo de detección de rostros llamando a[DescribeCollection \(p. 556\)](#).

Type: Cadena

Valores válidos: ROTATE\_0 | ROTATE\_90 | ROTATE\_180 | ROTATE\_270

[UnindexedFaces \(p. 647\)](#)

Matriz de rostros detectados en la imagen pero no indexados. No se indexaron porque el filtro de calidad los identificó como de baja calidad, o el[MaxFaces](#)el parámetro request los filtró. Para utilizar el filtro de calidad, especifique la[QualityFilter](#)parámetro de solicitud.

Type: Matriz de[UnindexedFace \(p. 842\)](#)objects

## Errores

[AccessDeniedException](#)

No tiene autorización para realizar la acción.

Código de estado HTTP: 400

[ImageTooLargeException](#)

La imagen de entrada tamaño supera el límite permitido. Si estás llamando[DetectProtectiveEquipment \(p. 592\)](#), el tamaño o la resolución de la imagen supera el límite permitido. Para obtener más información, consulte [Directrices y cuotas de Amazon Rekognition \(p. 847\)](#).

Código de estado HTTP: 400

**InternalServerError**

Amazon Lex ha tenido un problema de servicio. Pruebe la llamada de nuevo.

Código de estado HTTP: 500

**InvalidImageFormatException**

No se admite el formato de imagen proporcionado.

Código de estado HTTP: 400

**InvalidArgumentException**

El parámetro de entrada infringió una restricción. Valide el parámetro antes de llamar a la operación de la API de nuevo.

Código de estado HTTP: 400

**InvalidS3ObjectException**

Amazon Rekognition no puede obtener acceso al objeto de S3 especificado en la solicitud.

Código de estado HTTP: 400

**ProvisionedThroughputExceededException**

El número de solicitudes ha superado su límite de rendimiento. Si quieres aumentar este límite, ponte en contacto con Amazon Rekognition.

Código de estado HTTP: 400

**ResourceNotFoundException**

No se encuentra el recurso especificado en la solicitud.

Código de estado HTTP: 400

**ServiceQuotaExceededException**

El tamaño del recurso supera el límite permitido. Para obtener más información, consulte [Directrices y cuotas de Amazon Rekognition \(p. 847\)](#).

Código de estado HTTP: 400

**ThrottlingException**

Amazon Lex no puede procesar temporalmente la solicitud. Pruebe la llamada de nuevo.

Código de estado HTTP: 500

## Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- [AWS Command Line Interface](#)
- [SDK de AWS para .NET](#)
- [AWS SDK para C++](#)
- [AWS SDK para Go](#)
- [AWSSDK para Java V2](#)
- [AWS SDK para JavaScript](#)

- [SDK de AWS para PHP V3](#)
- [SDK de AWS para Python](#)
- [SDK de AWS para Ruby V3](#)

## ListCollections

Devuelve la lista de ID de colecciones de su cuenta. Si el resultado se trunca, la respuesta también proporciona un `NextToken` que puede utilizar en la solicitud subsiguiente para obtener el siguiente conjunto de ID de colecciones.

Para ver un ejemplo, consulte [Listado de colecciones \(p. 192\)](#).

Esta operación requiere permisos para realizar la acción `rekognition:ListCollections`.

### Sintaxis de la solicitud

```
{
 "MaxResults": number,
 "NextToken": "string"
}
```

### Parámetros de solicitud

La solicitud acepta los siguientes datos en formato JSON.

#### [MaxResults \(p. 653\)](#)

Número máximo de ID de colecciones a devolver.

Type: Entero

Rango válido: Valor mínimo de 0. Valor máximo de 4 096 caracteres.

Obligatorio: No

#### [NextToken \(p. 653\)](#)

Token de paginación de la respuesta anterior.

Type: Cadena

Restricciones de longitud: La longitud máxima es de 255 caracteres.

Obligatorio: No

### Sintaxis de la respuesta

```
{
 "CollectionIds": ["string"],
 "FaceModelVersions": ["string"],
 "NextToken": "string"
}
```

### Elementos de respuesta

Si la acción se realiza correctamente, el servicio devuelve una respuesta HTTP 200.

El servicio devuelve los datos siguientes en formato JSON.

#### [CollectionIds \(p. 653\)](#)

Una matriz de ID de colecciones.

Type: Matriz de cadenas

Restricciones de longitud: Longitud mínima de 1. La longitud máxima es de 255 caracteres.

Patrón: [ a-zA-Z0-9\_.\-\+]<sup>+</sup>

#### [FaceModelVersions \(p. 653\)](#)

Números de versión de los modelos de detección de rostros asociados a las colecciones de la matrizCollectionIds. Por ejemplo, el valor de FaceModelVersions[2] es el número de versión del modelo de detección de rostros utilizado por la colección enCollectionId[2].

Type: Matriz de cadenas

#### [NextToken \(p. 653\)](#)

Si el resultado se trunca, la respuesta proporciona unNextTokenque puede utilizar en la solicitud subsiguiente para obtener el siguiente conjunto de ID de colecciones.

Type: Cadena

Restricciones de longitud: La longitud máxima es de 255 caracteres.

## Errores

### AccessDeniedException

No tiene autorización para realizar la acción.

Código de estado HTTP: 400

### InternalServerError

Amazon Lex ha tenido un problema de servicio. Pruebe la llamada de nuevo.

Código de estado HTTP: 500

### InvalidPaginationTokenException

El token de paginación de la solicitud no es válido.

Código de estado HTTP: 400

### InvalidParameterException

El parámetro de entrada infringió una restricción. Valide el parámetro antes de llamar a la operación de la API de nuevo.

Código de estado HTTP: 400

### ProvisionedThroughputExceededException

El número de solicitudes ha superado su límite de rendimiento. Si necesita aumentar este límite, póngase en contacto con Amazon Rekognition.

Código de estado HTTP: 400

### ResourceNotFoundException

No se encuentra el recurso especificado en la solicitud.

Código de estado HTTP: 400

### ThrottlingException

Amazon Lex no puede procesar temporalmente la solicitud. Pruebe la llamada de nuevo.

Código de estado HTTP: 500

## Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- [AWS Command Line Interface](#)
- [SDK de AWS para .NET](#)
- [AWS SDK para C++](#)
- [AWS SDK para Go](#)
- [AWSSDK para Java V2](#)
- [AWS SDK para JavaScript](#)
- [SDK de AWS para PHP V3](#)
- [SDK de AWS para Python](#)
- [SDK de AWS para Ruby V3](#)

## ListDatasetEntries

Muestra las entradas (imágenes) de un conjunto de datos. Una entrada es una línea JSON que contiene la información de una sola imagen, incluida la ubicación de la imagen, las etiquetas asignadas y los cuadros delimitadores de ubicación de objetos. Para obtener más información, consulte [Creación de un archivo de manifiesto](#).

Las líneas JSON de la respuesta incluyen información sobre errores no terminales encontrados en el conjunto de datos. Los errores que no pertenecen a la terminal de `errors` se listan dentro de cada línea JSON. La misma información se informa en los manifiestos de resultados de validación de formación y pruebas que Amazon Rekognition Custom Labels crea durante la formación de modelos.

Puede filtrar la respuesta de diversas formas, como elegir qué etiquetas devolver y devolver líneas JSON creadas después de una fecha específica.

Esta operación requiere permisos para realizar la acción `rekognition:ListDatasetEntries`.

## Sintaxis de la solicitud

```
{
 "ContainsLabels": ["string"],
 "DatasetArn": "string",
 "HasErrors": boolean,
 "Labeled": boolean,
 "MaxResults": number,
 "NextToken": "string",
 "SourceRefContains": "string"
}
```

## Parámetros de solicitud

La solicitud acepta los siguientes datos en formato JSON.

### ContainsLabels ([p. 656](#))

Especifica un filtro de etiquetas para la respuesta. La respuesta incluye una entrada solo si una o más de las etiquetas de `ContainsLabels` en la entrada.

Type: Matriz de cadenas

Miembros de la matriz: Número mínimo de 1 elemento. Número máximo de 10 elementos.

Restricciones de longitud: Longitud mínima de 1. La longitud máxima es de 255 caracteres.

Patrón: `.{1,}`

Obligatorio: No

### DatasetArn ([p. 656](#))

El Nombre de recurso de Amazon (ARN) del conjunto de datos que desea utilizar.

Type: Cadena

Restricciones de longitud: Longitud mínima de 20. La longitud máxima es de 2048 caracteres.

Patrón: `(^arn:[a-z\d-]+:rekognition:[a-z\d-]+:\d{12}:project\/[a-zA-Z0-9_.\-\_]{1,255}\dataset\/(train|test)\/[0-9]+\$)`

Obligatorio: Sí

[HasErrors \(p. 656\)](#)

Especifica un filtro de errores para la respuesta. Especifique `.True`para incluir únicamente las entradas que tienen errores.

Type: Booleano

Obligatorio: No

[Labeled \(p. 656\)](#)

Especifique `.true`para obtener solo las líneas JSON en las que la imagen está etiquetada.

Especifique `.false`para obtener solo las líneas JSON en las que la imagen no está etiquetada. Si no especifica `Labeled`, `ListDatasetEntries`devuelve Líneas JSON para imágenes etiquetadas y sin etiquetar.

Type: Booleano

Obligatorio: No

[MaxResults \(p. 656\)](#)

El número máximo de resultados que devolver por llamada paginada. El valor más grande que puede especificar es 100. Si especifica un valor mayor que 100, se produce un error de `ValidationException`. El valor predeterminado es 100.

Type: Entero

Rango válido: Valor mínimo de 1. Valor máximo de 100.

Obligatorio: No

[NextToken \(p. 656\)](#)

Si la respuesta anterior estaba incompleta (porque hay más resultados que recuperar), Amazon Rekognition Custom Labels devuelve un token de paginación en la respuesta. Puede usar este token de paginación para recuperar el siguiente grupo de resultados.

Type: Cadena

Restricciones de longitud: La longitud máxima es de 1024 caracteres.

Obligatorio: No

[SourceRefContains \(p. 656\)](#)

Si se especifica `ListDatasetEntries` solo devuelve líneas JSON donde el valor `deSourceRefContains` forma parte de la `source-ref`. La `source-ref` campo contiene la ubicación de Amazon S3 de la imagen. Puede usar `SourceRefContains` para tareas tales como obtener la línea JSON para una sola imagen o obtener líneas JSON para todas las imágenes de una carpeta específica.

Para obtener más información, consulte [Creación de un archivo de manifiesto](#).

Para obtener más información, consulte Creación de un archivo de manifiesto en el Guía para desarrolladores de etiquetas personalizadas de Amazon Rekognition.

Type: Cadena

Restricciones de longitud: Longitud mínima de 1. La longitud máxima es de 2048 caracteres.

Patrón: `.*\S.*`

Obligatorio: No

## Sintaxis de la respuesta

```
{
 "DatasetEntries": ["string"],
 "NextToken": "string"
}
```

## Elementos de respuesta

Si la acción se realiza correctamente, el servicio devuelve una respuesta HTTP 200.

El servicio devuelve los datos siguientes en formato JSON.

### [DatasetEntries \(p. 658\)](#)

Lista de entradas (imágenes) del conjunto de datos.

Type: Matriz de cadenas

Restricciones de longitud: Longitud mínima de 1. La longitud máxima es de 100000 caracteres.

Patrón: ^\{.\*\}\$

### [NextToken \(p. 658\)](#)

Si la respuesta anterior estaba incompleta (porque hay más resultados que recuperar), Amazon Rekognition Custom Labels devuelve un token de paginación en la respuesta. Puede usar este token de paginación para recuperar el siguiente grupo de resultados.

Type: Cadena

Restricciones de longitud: La longitud máxima es de 1024 caracteres.

## Errores

### [AccessDeniedException](#)

No tiene autorización para realizar la acción.

Código de estado HTTP: 400

### [InternalServerError](#)

Amazon Lex ha tenido un problema de servicio. Pruebe la llamada de nuevo.

Código de estado HTTP: 500

### [InvalidPaginationTokenException](#)

El token de paginación de la solicitud no es válido.

Código de estado HTTP: 400

### [InvalidArgumentException](#)

El parámetro de entrada infringió una restricción. Valide el parámetro antes de llamar a la operación de la API de nuevo.

Código de estado HTTP: 400

ProvisionedThroughputExceededException

El número de solicitudes ha superado su límite de rendimiento. Si necesita aumentar este límite, póngase en contacto con Amazon Rekognition.

Código de estado HTTP: 400

ResourceInUseException

El recurso especificado ya se está utilizando.

Código de estado HTTP: 400

ResourceNotFoundException

No se encuentra el recurso especificado en la solicitud.

Código de estado HTTP: 400

ResourceNotReadyException

El recurso solicitado no está listo. Por ejemplo, esta excepción se produce cuando `llamaDetectCustomLabels` con una versión de modelo que no está implementada.

Código de estado HTTP: 400

ThrottlingException

Amazon Lex no puede procesar temporalmente la solicitud. Pruebe la llamada de nuevo.

Código de estado HTTP: 500

## Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- [AWS Command Line Interface](#)
- [SDK de AWS para .NET](#)
- [AWS SDK para C++](#)
- [AWS SDK para Go](#)
- [AWSSDK para Java V2](#)
- [AWS SDK para JavaScript](#)
- [SDK de AWS para PHP V3](#)
- [SDK de AWS para Python](#)
- [SDK de AWS para Ruby V3](#)

## ListDatasetLabels

Enumera las etiquetas de un conjunto de datos. Las etiquetas personalizadas de Amazon Rekognition utilizan etiquetas para describir imágenes. Para obtener más información, consulte [Etiquetado de imágenes](#).

Esta operación requiere permisos para realizar la acción `rekognition:ListDatasetLabels`.

### Sintaxis de la solicitud

```
{
 "DatasetArn": "string",
 "MaxResults": number,
 "NextToken": "string"
}
```

### Parámetros de solicitud

La solicitud acepta los siguientes datos en formato JSON.

#### [DatasetArn](#) (p. 660)

El nombre de recurso de Amazon (ARN) del conjunto de datos que desea utilizar.

Type: Cadena

Restricciones de longitud: Longitud mínima de 20. La longitud máxima es de 2048 caracteres.

Patrón: (^arn:[a-z\d-]+:rekognition:[a-z\d-]+\:\d{12}:project\/[a-zA-Z0-9\_.\-\\_]{1,255}\dataset\/(train|test)\/[0-9]+\\$)

Obligatorio: Sí

#### [MaxResults](#) (p. 660)

El número máximo de resultados que devolver por llamada paginada. El valor más grande que puede especificar es 100. Si especifica un valor mayor que 100, se produce un error de ValidationException. El valor predeterminado es 100.

Type: Entero

Rango válido: Valor mínimo de 1. Valor máximo de 100.

Obligatorio: No

#### [NextToken](#) (p. 660)

Si la respuesta anterior estaba incompleta (porque hay más resultados que recuperar), Amazon Rekognition Custom Labels devuelve un token de paginación en la respuesta. Puede utilizar este token de paginación para recuperar el siguiente grupo de resultados.

Type: Cadena

Restricciones de longitud: La longitud máxima es de 1024 caracteres.

Obligatorio: No

### Sintaxis de la respuesta

```
{
```

```
"DatasetLabelDescriptions": [
 {
 "LabelName": "string",
 "LabelStats": {
 "BoundingBoxCount": number,
 "EntryCount": number
 }
 }
,
 "NextToken": "string"
}
```

## Elementos de respuesta

Si la acción se realiza correctamente, el servicio devuelve una respuesta HTTP 200.

El servicio devuelve los datos siguientes en formato JSON.

### [DatasetLabelDescriptions \(p. 660\)](#)

Lista de las etiquetas del conjunto de datos.

Type: Matriz de[DatasetLabelDescription \(p. 757\)](#)objects

### [NextToken \(p. 660\)](#)

Si la respuesta anterior estaba incompleta (porque hay más resultados que recuperar), Amazon Rekognition Custom Labels devuelve un token de paginación en la respuesta. Puede utilizar este token de paginación para recuperar el siguiente grupo de resultados.

Type: Cadena

Restricciones de longitud: La longitud máxima es de 1024 caracteres.

## Errores

### [AccessDeniedException](#)

No tiene autorización para realizar la acción.

Código de estado HTTP: 400

### [InternalServerError](#)

Amazon Lex ha tenido un problema de servicio. Pruebe la llamada de nuevo.

Código de estado HTTP: 500

### [InvalidPaginationTokenException](#)

El token de paginación de la solicitud no es válido.

Código de estado HTTP: 400

### [InvalidParameterException](#)

El parámetro de entrada infringió una restricción. Valide el parámetro antes de llamar a la operación de la API de nuevo.

Código de estado HTTP: 400

### [ProvisionedThroughputExceededException](#)

El número de solicitudes ha superado su límite de rendimiento. Si necesita aumentar este límite, póngase en contacto con Amazon Rekognition.

Código de estado HTTP: 400  
`ResourceInUseException`

El recurso especificado ya se está utilizando.

Código de estado HTTP: 400  
`ResourceNotFoundException`

No se encuentra el recurso especificado en la solicitud.

Código de estado HTTP: 400  
`ResourceNotReadyException`

El recurso solicitado no está listo. Por ejemplo, esta excepción se produce cuando `llamaDetectCustomLabels` con una versión de modelo que no está implementada.

Código de estado HTTP: 400  
`ThrottlingException`

Amazon Lex no puede procesar temporalmente la solicitud. Pruebe la llamada de nuevo.

Código de estado HTTP: 500

## Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- [AWS Command Line Interface](#)
- [SDK de AWS para .NET](#)
- [AWS SDK para C++](#)
- [AWS SDK para Go](#)
- [AWSSDK para Java V2](#)
- [AWS SDK para JavaScript](#)
- [SDK de AWS para PHP V3](#)
- [SDK de AWS para Python](#)
- [SDK de AWS para Ruby V3](#)

## ListFaces

Devuelve los metadatos de los rostros de la colección especificada. Estos metadatos incluyen información como las coordenadas del contorno, la confianza (de que el contorno contenga un rostro) y el identificador de rostro. Para ver un ejemplo, consulte [Listado de rostros en una colección \(p. 216\)](#).

Esta operación requiere permisos para realizar la acción `rekognition:ListFaces`.

### Sintaxis de la solicitud

```
{
 "CollectionId": "string",
 "MaxResults": number,
 "NextToken": "string"
}
```

### Parámetros de solicitud

La solicitud acepta los siguientes datos en formato JSON.

#### [CollectionId \(p. 663\)](#)

ID de la colección desde la que se van a enumerar las caras.

Type: Cadena

Restricciones de longitud: Longitud mínima de 1. La longitud máxima es de 255 caracteres.

Patrón: [a-zA-Z0-9\_.\-\+]+

Obligatorio: Sí

#### [MaxResults \(p. 663\)](#)

Número máximo de rostros a devolver.

Type: Entero

Rango válido: Valor mínimo de 0. Valor máximo de 4 096 caracteres.

Obligatorio: No

#### [NextToken \(p. 663\)](#)

Si la respuesta anterior estaba incompleta (porque hay más datos que recuperar), Amazon Rekognition devuelve un token de paginación en la respuesta. Puede utilizar este token de paginación para recuperar el siguiente conjunto de rostros.

Type: Cadena

Restricciones de longitud: La longitud máxima es de 255 caracteres.

Obligatorio: No

### Sintaxis de la respuesta

```
{
 "FaceModelVersion": "string",
```

```
"Faces": [
 {
 "BoundingBox": {
 "Height": number,
 "Left": number,
 "Top": number,
 "Width": number
 },
 "Confidence": number,
 "ExternalImageId": "string",
 "FaceId": "string",
 "ImageId": "string",
 "IndexFacesModelVersion": "string"
 }
],
"NextToken": "string"
}
```

## Elementos de respuesta

Si la acción se realiza correctamente, el servicio devuelve una respuesta HTTP 200.

El servicio devuelve los datos siguientes en formato JSON.

### [FaceModelVersion \(p. 663\)](#)

Número de versión del modelo de detección de rostros asociado a la colección de entrada (`CollectionId`).

Type: Cadena

### [Faces \(p. 663\)](#)

Una matriz de objetos `Face`.

Type: Matriz de [Face \(p. 771\)](#)objects

### [NextToken \(p. 663\)](#)

Si la respuesta se trunca, Amazon Rekognition devuelve este token que puede utilizar en la solicitud posterior para recuperar el siguiente conjunto de rostros.

Type: Cadena

## Errores

### [AccessDeniedException](#)

No tiene autorización para realizar la acción.

Código de estado HTTP: 400

### [InternalServerError](#)

Amazon Lex ha tenido un problema de servicio. Pruebe la llamada de nuevo.

Código de estado HTTP: 500

### [InvalidPaginationTokenException](#)

El token de paginación de la solicitud no es válido.

Código de estado HTTP: 400

#### InvalidArgumentException

El parámetro de entrada infringió una restricción. Valide el parámetro antes de llamar a la operación de la API de nuevo.

Código de estado HTTP: 400

#### ProvisionedThroughputExceededException

El número de solicitudes ha superado su límite de rendimiento. Si necesitas aumentar este límite, ponte en contacto con Amazon Rekognition.

Código de estado HTTP: 400

#### ResourceNotFoundException

No se encuentra el recurso especificado en la solicitud.

Código de estado HTTP: 400

#### ThrottlingException

Amazon Lex no puede procesar temporalmente la solicitud. Pruebe la llamada de nuevo.

Código de estado HTTP: 500

## Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- [AWS Command Line Interface](#)
- [SDK de AWS para .NET](#)
- [AWS SDK para C++](#)
- [AWS SDK para Go](#)
- [AWSSDK para Java V2](#)
- [AWS SDK para JavaScript](#)
- [SDK de AWS para PHP V3](#)
- [SDK de AWS para Python](#)
- [SDK de AWS para Ruby V3](#)

## ListStreamProcessors

Obtiene una lista de los procesadores de streaming que ha creado con[CreateStreamProcessor \(p. 537\)](#).

### Sintaxis de la solicitud

```
{
 "MaxResults": number,
 "NextToken": string
}
```

### Parámetros de solicitud

La solicitud acepta los siguientes datos en formato JSON.

#### [MaxResults \(p. 666\)](#)

El número máximo de los procesadores de streaming que desea que Amazon Rekognition Video devuelva en la respuesta. El valor predeterminado es 1000.

Type: Entero

Rango válido: Valor mínimo de 1.

Obligatorio: No

#### [NextToken \(p. 666\)](#)

Si la respuesta anterior estaba incompleta (porque hay más procesadores de transmisión que recuperar), Amazon Rekognition Video devuelve un token de paginación en la respuesta. Puede utilizar este token de paginación para recuperar el siguiente conjunto de procesadores de streaming.

Type: Cadena

Restricciones de longitud: La longitud máxima es de 255 caracteres.

Obligatorio: No

### Sintaxis de la respuesta

```
{
 "NextToken": string,
 "StreamProcessors": [
 {
 "Name": string,
 "Status": string
 }
]
}
```

### Elementos de respuesta

Si la acción se realiza correctamente, el servicio devuelve una respuesta HTTP 200.

El servicio devuelve los datos siguientes en formato JSON.

### [NextToken \(p. 666\)](#)

Si la respuesta se trunca, Amazon Rekognition Video devuelve este token que puede utilizar en la solicitud posterior para recuperar el siguiente conjunto de procesadores de transmisión.

Type: Cadena

Restricciones de longitud: La longitud máxima es de 255 caracteres.

### [StreamProcessors \(p. 666\)](#)

Lista de los procesadores de streaming que ha creado.

Type: Matriz de[StreamProcessor \(p. 828\)](#)objects

## Errores

### AccessDeniedException

No tiene autorización para realizar la acción.

Código de estado HTTP: 400

### InternalServerError

Amazon Lex ha tenido un problema de servicio. Pruebe la llamada de nuevo.

Código de estado HTTP: 500

### InvalidPaginationTokenException

El token de paginación de la solicitud no es válido.

Código de estado HTTP: 400

### InvalidParameterException

El parámetro de entrada infringió una restricción. Valide el parámetro antes de llamar a la operación de la API de nuevo.

Código de estado HTTP: 400

### ProvisionedThroughputExceededException

El número de solicitudes ha superado su límite de rendimiento. Si necesita aumentar este límite, póngase en contacto con Amazon Rekognition.

Código de estado HTTP: 400

### ThrottlingException

Amazon Lex no puede procesar temporalmente la solicitud. Pruebe la llamada de nuevo.

Código de estado HTTP: 500

## Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- [AWS Command Line Interface](#)
- [SDK de AWS para .NET](#)

- [AWS SDK para C++](#)
- [AWS SDK para Go](#)
- [AWSSDK para Java V2](#)
- [AWS SDK para JavaScript](#)
- [SDK de AWS para PHP V3](#)
- [SDK de AWS para Python](#)
- [SDK de AWS para Ruby V3](#)

## ListTagsForResource

Devuelve una lista de etiquetas de una colección de Amazon Rekognition, un procesador de secuencias o un modelo de etiquetas personalizadas.

Esta operación requiere permisos para realizar la acción `rekognition:ListTagsForResource`.

### Sintaxis de la solicitud

```
{
 "ResourceArn": "string"
}
```

### Parámetros de solicitud

La solicitud acepta los siguientes datos en formato JSON.

#### [ResourceArn \(p. 669\)](#)

Nombre de recurso de Amazon (ARN) del modelo, la colección de o el procesador de flujo de que contiene las etiquetas de las que deseé enumerar.

Type: Cadena

Restricciones de longitud: Longitud mínima de 20. La longitud máxima es de 2048 caracteres.

Obligatorio: Sí

### Sintaxis de la respuesta

```
{
 "Tags": {
 "string" : "string"
 }
}
```

### Elementos de respuesta

Si la acción se realiza correctamente, el servicio devuelve una respuesta HTTP 200.

El servicio devuelve los datos siguientes en formato JSON.

#### [Tags \(p. 669\)](#)

Una lista de etiquetas clave-valor asignadas al recurso.

Type: Asignación de cadena a cadena

Entradas de mapa: El número mínimo es 0 elementos. Número máximo de 200 elementos.

Restricciones de longitud clave: Longitud mínima de 1. La longitud máxima es de 128 caracteres.

Patrón de clave: ^(?!aws:)[\p{L}\p{Z}\p{N}\_.:/+=\-\@]\*\$

Restricciones de longitud de valor: Longitud mínima de 0. La longitud máxima es de 256 caracteres.

Patrón de valor:`^([ \p{L}\p{Z}\p{N}_.:+=\-\@]* )$`

## Errores

### AccessDeniedException

No tiene autorización para realizar la acción.

Código de estado HTTP: 400

### InternalServerError

Amazon Lex ha tenido un problema de servicio. Pruebe la llamada de nuevo.

Código de estado HTTP: 500

### InvalidParameterException

El parámetro de entrada infringió una restricción. Valide el parámetro antes de llamar a la operación de la API de nuevo.

Código de estado HTTP: 400

### ProvisionedThroughputExceededException

El número de solicitudes ha superado su límite de rendimiento. Si necesita aumentar este límite, póngase en contacto con Amazon Rekognition.

Código de estado HTTP: 400

### ResourceNotFoundException

No se encuentra el recurso especificado en la solicitud.

Código de estado HTTP: 400

### ThrottlingException

Amazon Lex no puede procesar temporalmente la solicitud. Pruebe la llamada de nuevo.

Código de estado HTTP: 500

## Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- [AWS Command Line Interface](#)
- [SDK de AWS para .NET](#)
- [AWS SDK para C++](#)
- [AWS SDK para Go](#)
- [AWSSDK para Java V2](#)
- [AWS SDK para JavaScript](#)
- [SDK de AWS para PHP V3](#)
- [SDK de AWS para Python](#)
- [SDK de AWS para Ruby V3](#)

## RecognizeCelebrities

Devuelve una matriz de famosos reconocidos en la imagen de entrada. Para obtener más información, consulte [Reconocimiento de famosos \(p. 275\)](#).

RecognizeCelebritiesdevuelve las 64 caras más grandes de la imagen. Enumera las celebridades reconocidas en elCelebrityFacesmatriz y cualquier cara no reconocida en elUnrecognizedFacesmatriz. RecognizeCelebritiesno devuelve celebridades cuyos rostros no se encuentran entre las 64 caras más grandes de la imagen.

Por cada celebridad reconocida, RecognizeCelebritiesdevuelve un objetoCelebrityobjeto. LaCelebrityobjeto contiene el nombre de la celebridad, el ID, los enlaces URL a información adicional, la confianza de coincidencia y unComparedFaceobjeto que puedes usar para localizar el rostro de la celebridad en la imagen.

Amazon Rekognition no conserva información sobre las imágenes en las que se ha reconocido una celebridad. La aplicación debe almacenar esta información y utilizar laCelebrityPropiedad ID como identificador único para la celebridad. Si no almacenas el nombre de la celebridad o las URL de información adicional devueltas porRecognizeCelebrities, necesitará el ID para identificar a la celebridad en una llamada al[GetCelebrityInfo \(p. 602\)](#).

Se pasa la imagen de entrada como bytes de imagen codificados en base64 o como referencia a una imagen de un bucket de Amazon S3. Si utiliza la CLI de AWS para llamar a las operaciones de Amazon Rekognition, no es posible pasar bytes de imágenes. La imagen debe ser un archivo de formato PNG o JPEG.

Para ver un ejemplo, consulte [Reconocimiento de famosos en una imagen \(p. 276\)](#).

Esta operación requiere permisos para ejecutar `rekognition:RecognizeCelebrities`.

## Sintaxis de la solicitud

```
{
 "Image": {
 "Bytes": blob,
 "S3Object": {
 "Bucket": "string",
 "Name": "string",
 "Version": "string"
 }
 }
}
```

## Parámetros de solicitud

La solicitud acepta los siguientes datos en formato JSON.

### [Image \(p. 671\)](#)

La imagen de entrada como bytes codificados en base64 o un objeto S3. Si utiliza la CLI de AWS para llamar a las operaciones de Amazon Rekognition, no es posible pasar bytes de imágenes codificadas en base 64.

Si utiliza un SDK de AWS para llamar a Amazon Rekognition, es posible que no tenga que codificar en base 64 bytes de imagen pasados mediante laBytes. Para obtener más información, consulte [Especificaciones de imágenes \(p. 29\)](#).

Tipo: objeto [Image](#) (p. 786)

Obligatorio: Sí

## Sintaxis de la respuesta

```
{
 "CelebrityFaces": [
 {
 "Face": {
 "BoundingBox": {
 "Height": number,
 "Left": number,
 "Top": number,
 "Width": number
 },
 "Confidence": number,
 "Emotions": [
 {
 "Confidence": number,
 "Type": "string"
 }
],
 "Landmarks": [
 {
 "Type": "string",
 "X": number,
 "Y": number
 }
],
 "Pose": {
 "Pitch": number,
 "Roll": number,
 "Yaw": number
 },
 "Quality": {
 "Brightness": number,
 "Sharpness": number
 },
 "Smile": {
 "Confidence": number,
 "Value": boolean
 }
 },
 "Id": "string",
 "KnownGender": {
 "Type": "string"
 },
 "MatchConfidence": number,
 "Name": "string",
 "Urls": ["string"]
 }
],
 "OrientationCorrection": "string",
 "UnrecognizedFaces": [
 {
 "BoundingBox": {
 "Height": number,
 "Left": number,
 "Top": number,
 "Width": number
 },
 "Confidence": number,
 "Type": "string"
 }
]
}
```

```
 "Emotions": [
 {
 "Confidence": number,
 "Type": "string"
 }
],
 "Landmarks": [
 {
 "Type": "string",
 "X": number,
 "Y": number
 }
],
 "Pose": {
 "Pitch": number,
 "Roll": number,
 "Yaw": number
 },
 "Quality": {
 "Brightness": number,
 "Sharpness": number
 },
 "Smile": {
 "Confidence": number,
 "Value": boolean
 }
 }
]
```

## Elementos de respuesta

Si la acción se realiza correctamente, el servicio devuelve una respuesta HTTP 200.

El servicio devuelve los datos siguientes en formato JSON.

### CelebrityFaces (p. 672)

Detalles sobre cada celebridad que se encuentra en la imagen. Amazon Rekognition puede detectar un máximo de 64 celebridades en una imagen. Cada objeto de celebridad incluye los siguientes atributos:Face,Confidence,Emotions,Landmarks,Pose,Quality,Smile,Id,KnownGender,MatchConfidence

Type: Matriz deCelebrity (p. 742)objects

### OrientationCorrection (p. 672)

#### Note

La Support con la estimación de la orientación de la imagen mediante el campo OrientationCorrection ha cesado en agosto de 2021. Los valores devueltos para este campo incluidos en una respuesta de API siempre serán NULL.

Orientación de la imagen de entrada (dirección antihorario). Si la aplicación muestra la imagen, puede utilizar este valor para corregir la orientación. Las coordenadas del cuadro delimitador devuelven enCelebrityFacesyUnrecognizedFacesrepresenta las ubicaciones de las caras antes de corregir la orientación de la imagen.

#### Note

Si la imagen de entrada está en formato.jpeg, podría contener metadatos de imagen intercambiable (Exif) que incluyen la orientación de la imagen. Si es así, y los metadatos Exif de la imagen de entrada llenan el campo de orientación, el valor deOrientationCorrectiones nulo. LaCelebrityFacesyUnrecognizedFacesLas

coordenadas del cuadro delimitador representan las ubicaciones de las caras después de utilizar los metadatos Exif para corregir la orientación de la imagen. Las imágenes en formato .png no contienen metadatos Exif.

Type: Cadena

Valores válidos: ROTATE\_0 | ROTATE\_90 | ROTATE\_180 | ROTATE\_270

[UnrecognizedFaces \(p. 672\)](#)

Detalles sobre cada cara no reconocida de la imagen.

Type: Matriz de[ComparedFace \(p. 747\)](#)objects

## Errores

AccessDeniedException

No tiene autorización para realizar la acción.

Código de estado HTTP: 400

ImageTooLargeException

La imagen de entrada tamaño supera el límite permitido. Si estás llamando[DetectProtectiveEquipment \(p. 592\)](#), el tamaño o la resolución de imágenes superan el límite permitido. Para obtener más información, consulte [Directrices y cuotas de Amazon Rekognition \(p. 847\)](#).

Código de estado HTTP: 400

InternalServerError

Amazon Lex ha tenido un problema de servicio. Pruebe la llamada de nuevo.

Código de estado HTTP: 500

InvalidImageFormatException

No se admite el formato de imagen proporcionado.

Código de estado HTTP: 400

InvalidImageFormatException

No se admite el formato de imagen proporcionado.

Código de estado HTTP: 400

InvalidParameterException

El parámetro de entrada infringió una restricción. Valide el parámetro antes de llamar a la operación de la API de nuevo.

Código de estado HTTP: 400

InvalidS3ObjectException

Amazon Rekognition no puede obtener acceso al objeto de S3 especificado en la solicitud.

Código de estado HTTP: 400

ProvisionedThroughputExceededException

El número de solicitudes ha superado su límite de rendimiento. Si necesita aumentar este límite, póngase en contacto con Amazon Rekognition.

Código de estado HTTP: 400  
ThrottlingException

Amazon Lex no puede procesar temporalmente la solicitud. Pruebe la llamada de nuevo.

Código de estado HTTP: 500

## Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- [AWS Command Line Interface](#)
- [SDK de AWS para .NET](#)
- [AWS SDK para C++](#)
- [AWS SDK para Go](#)
- [AWSSDK para Java V2](#)
- [AWS SDK para JavaScript](#)
- [SDK de AWS para PHP V3](#)
- [SDK de AWS para Python](#)
- [SDK de AWS para Ruby V3](#)

## SearchFaces

Para un ID de rostro de entrada determinado, busca rostros coincidentes en la colección a la que pertenece la cara. Obtienes un ID de rostro cuando añades una cara a la colección mediante [IndexFaces \(p. 644\)](#). La operación compara las entidades de la cara de entrada con las caras de la colección especificada.

### Note

También puede buscar caras sin indexar caras mediante la [SearchFacesByImage](#).

La respuesta de la operación devuelve una matriz de caras que coinciden, ordenadas por puntuación de similitud con la mayor similitud primero. Más concretamente, se trata de una matriz de metadatos para cada coincidencia facial que se encuentra. Junto con los metadatos, la respuesta también incluye un confidence valor para cada coincidencia de caras, lo que indica la confianza de que la cara específica coincide con la cara de entrada.

Para ver un ejemplo, consulte [Búsqueda de un rostro mediante su ID de cara \(p. 224\)](#).

Esta operación requiere permisos para realizar la acción `rekognition:SearchFaces`.

## Sintaxis de la solicitud

```
{
 "CollectionId": "string",
 "FaceId": "string",
 "FaceMatchThreshold": number,
 "MaxFaces": number
}
```

## Parámetros de solicitud

La solicitud acepta los siguientes datos en formato JSON.

### CollectionId ([p. 676](#))

ID de la colección a la que pertenece la cara.

Type: Cadena

Restricciones de longitud: Longitud mínima de 1. La longitud máxima es de 255 caracteres.

Patrón: [ a-zA-Z0-9\_.\-\+ ]+

Obligatorio: Sí

### Facelid ([p. 676](#))

Identificador de una cara para encontrar coincidencias en la colección.

Type: Cadena

Patrón: [ 0-9a-f ]{8}-[ 0-9a-f ]{4}-[ 0-9a-f ]{4}-[ 0-9a-f ]{4}-[ 0-9a-f ]{12}

Obligatorio: Sí

### FaceMatchThreshold ([p. 676](#))

Valor opcional que especifica la confianza mínima en la coincidencia facial que se va a devolver. Por ejemplo, no devuelvas ningún partido en el que la confianza en los partidos sea inferior al 70%. El valor predeterminado es del 80%.

Type: Float

Rango válido: Valor mínimo de 0. Valor máximo de 100.

Obligatorio: No

[MaxFaces \(p. 676\)](#)

Número máximo de rostros que se devuelve. La operación devuelve el número máximo de caras con la mayor confianza en la partida.

Type: Entero

Rango válido: Valor mínimo de 1. Valor máximo de 4 096 caracteres.

Obligatorio: No

## Sintaxis de la respuesta

```
{
 "FaceMatches": [
 {
 "Face": {
 "BoundingBox": {
 "Height": "number",
 "Left": "number",
 "Top": "number",
 "Width": "number"
 },
 "Confidence": "number",
 "ExternalImageId": "string",
 "FaceId": "string",
 "ImageId": "string",
 "IndexFacesModelVersion": "string"
 },
 "Similarity": "number"
 }
],
 "FaceModelVersion": "string",
 "SearchedFaceId": "string"
}
```

## Elementos de respuesta

Si la acción se realiza correctamente, el servicio devuelve una respuesta HTTP 200.

El servicio devuelve los datos siguientes en formato JSON.

[FaceMatches \(p. 677\)](#)

Matriz de caras que coinciden con la cara de entrada, junto con la confianza en la partida.

Type: Matriz de[FaceMatch \(p. 777\)](#)objects

[FaceModelVersion \(p. 677\)](#)

Número de versión del modelo de detección de rostros asociada a la colección de entrada ([CollectionId](#)).

Type: Cadena

### [SearchedFaceld \(p. 677\)](#)

ID de la cara en la que se buscaron coincidencias en una colección.

Type: Cadena

Patrón: [0-9a-f]{8}-[0-9a-f]{4}-[0-9a-f]{4}-[0-9a-f]{4}-[0-9a-f]{12}

## Errores

### AccessDeniedException

No tiene autorización para realizar la acción.

Código de estado HTTP: 400

### InternalServerError

Amazon Lex ha tenido un problema de servicio. Pruebe la llamada de nuevo.

Código de estado HTTP: 500

### InvalidParameterException

El parámetro de entrada infringió una restricción. Valide el parámetro antes de llamar a la operación de la API de nuevo.

Código de estado HTTP: 400

### ProvisionedThroughputExceededException

El número de solicitudes ha superado su límite de rendimiento. Si necesita aumentar este límite, póngase en contacto con Amazon Rekognition.

Código de estado HTTP: 400

### ResourceNotFoundException

No se encuentra el recurso especificado en la solicitud.

Código de estado HTTP: 400

### ThrottlingException

Amazon Lex no puede procesar temporalmente la solicitud. Pruebe la llamada de nuevo.

Código de estado HTTP: 500

## Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- [AWS Command Line Interface](#)
- [SDK de AWS para .NET](#)
- [AWS SDK para C++](#)
- [AWS SDK para Go](#)
- [AWSSDK para Java V2](#)
- [AWS SDK para JavaScript](#)
- [SDK de AWS para PHP V3](#)

- [SDK de AWS para Python](#)
- [SDK de AWS para Ruby V3](#)

## SearchFacesByImage

Para una imagen de entrada determinada, detecta primero el rostro de mayor tamaño en la imagen y después busca rostros coincidentes en la colección especificada. La operación compara las entidades de la cara de entrada con las caras de la colección especificada.

### Note

Para buscar todas las caras de una imagen de entrada, primero puede llamar [allIndexFaces \(p. 644\)](#) y, a continuación, utilizar los identificadores de cara devueltos en las llamadas posteriores al [SearchFaces \(p. 676\)](#).

También puede llamar a `laDetectFaces` y utilizar los cuadros delimitadores de la respuesta para realizar recortes de cara, que luego puede pasar al `SearchFacesByImage`.

Se pasa la imagen de entrada como bytes de imagen codificados en base64 o como referencia a una imagen de un bucket de Amazon S3. Si utiliza la AWS CLI para llamar a las operaciones de Amazon Rekognition, no es posible pasar bytes de imágenes. La imagen debe ser un archivo de formato PNG o JPEG.

La respuesta devuelve una matriz de caras que coinciden, ordenadas por puntuación de similitud con la mayor similitud primero. Más concretamente, se trata de una matriz de metadatos para cada coincidencia de cara encontrada. Junto con los metadatos, la respuesta también incluye `unsimilarity` que indica cuán similar es la cara a la cara de entrada. En la respuesta, la operación también devuelve el cuadro delimitador (y un nivel de confianza que el cuadro delimitador contiene una cara) de la cara que Amazon Rekognition utilizó para la imagen de entrada.

Si no se detectan caras en la imagen de entrada, `SearchFacesByImage` devuelve `unInvalidParameterException`.

Para ver un ejemplo, consulte [Búsqueda de un rostro mediante una imagen \(p. 229\)](#).

La `QualityFilter` el parámetro de entrada le permite filtrar las caras detectadas que no cumplen con la barra de calidad requerida. La barra de calidad se basa en una variedad de casos de uso comunes. Usar `QualityFilter` para configurar la barra de calidad para el filtrado especificando `LOW`, `MEDIUM`, o bien `HIGH`. Si no desea filtrar las caras detectadas, especifique `NONE`. El valor predeterminado es `NONE`.

### Note

Para utilizar el filtrado según la calidad, necesita una colección asociada a la versión 3 del modelo de rostros o posterior. Para obtener la versión del modelo de rostros asociado a una colección, llame a [DescribeCollection \(p. 556\)](#).

Esta operación requiere permisos para realizar la acción `rekognition:SearchFacesByImage`.

## Sintaxis de la solicitud

```
{
 "CollectionId": "string",
 "FaceMatchThreshold": number,
 "Image": {
 "Bytes": blob,
 "S3Object": {
 "Bucket": "string",
 "Name": "string",
 "Version": "string"
 }
 },
 "MaxFaces": number,
 "QualityFilter": "string"
```

}

## Parámetros de solicitud

La solicitud acepta los siguientes datos en formato JSON.

### [CollectionId](#) (p. 680)

ID de la colección que se va a buscar.

Type: Cadena

Limitaciones de longitud: Longitud mínima de 1. La longitud máxima es de 255 caracteres.

Patrón: [ a-zA-Z0-9\_.\-\+]+

Obligatorio: Sí

### [FaceMatchThreshold](#) (p. 680)

(Opcional) Especifica la confianza mínima en la coincidencia facial que se va a devolver. Por ejemplo, no devuelvas ninguna partida en la que la confianza en los partidos sea inferior al 70%. El valor predeterminado es del 80%.

Type: Float

Rango válido: Valor mínimo de 0. Valor máximo de 100.

Obligatorio: No

### [Image](#) (p. 680)

La imagen de entrada como bytes codificados en base64 o un objeto S3. Si utiliza la AWS CLI para llamar a las operaciones de Amazon Rekognition, no es posible pasar bytes de imágenes codificados en base64.

Si utiliza un SDK de AWS para llamar a Amazon Rekognition, es posible que no tenga que codificar en base 64 bytes de imagen pasados mediante laBytes. Para obtener más información, consulte [Especificaciones de imágenes](#) (p. 29).

Tipo: objeto [Image](#) (p. 786)

Obligatorio: Sí

### [MaxFaces](#) (p. 680)

Número máximo de rostros a devolver. La operación devuelve el número máximo de caras con la mayor confianza en la partida.

Type: Entero

Rango válido: Valor mínimo de 1. Valor máximo de 4 096 caracteres.

Obligatorio: No

### [QualityFilter](#) (p. 680)

Filtro que especifica una barra de calidad para determinar cuánto filtrado se realiza para identificar caras. Las caras filtradas no se buscan en la colección. Si especificas AUTO, Amazon Rekognition elige la barra de calidad. Si especificas LOW, MEDIUM, o bien HIGH, el filtrado elimina todas las caras que no cumplen con la barra de calidad elegida. La barra de calidad se basa en una variedad de casos de uso comunes. Las detecciones de baja calidad pueden ocurrir por diversas razones. Algunos ejemplos son un objeto que se identifica mal como cara, cara demasiado borrosa o cara con una pose demasiado

extrema para usarla. Si especificas `NONE`, no se realiza ningún filtrado. El valor predeterminado es `NONE`.

Para utilizar el filtrado según la calidad, la colección que utiliza debe estar asociada a la versión 3 del modelo de rostros o posterior.

Type: Cadena

Valores válidos: `NONE` | `AUTO` | `LOW` | `MEDIUM` | `HIGH`

Obligatorio: No

## Sintaxis de la respuesta

```
{
 "FaceMatches": [
 {
 "Face": {
 "BoundingBox": {
 "Height": number,
 "Left": number,
 "Top": number,
 "Width": number
 },
 "Confidence": number,
 "ExternalImageId": "string",
 "FaceId": "string",
 "ImageId": "string",
 "IndexFacesModelVersion": "string"
 },
 "Similarity": number
 }
],
 "FaceModelVersion": "string",
 "SearchedFaceBoundingBox": {
 "Height": number,
 "Left": number,
 "Top": number,
 "Width": number
 },
 "SearchedFaceConfidence": number
}
```

## Elementos de respuesta

Si la acción se realiza correctamente, el servicio devuelve una respuesta HTTP 200.

El servicio devuelve los datos siguientes en formato JSON.

### [FaceMatches \(p. 682\)](#)

Matriz de caras que coinciden con la cara de entrada, junto con la confianza en la partida.

Type: Matriz de [FaceMatch \(p. 777\)](#) objects

### [FaceModelVersion \(p. 682\)](#)

Número de versión del modelo de detección de rostros asociado a la colección de entrada (`CollectionId`).

Type: Cadena

### [SearchedFaceBoundingBox \(p. 682\)](#)

Cuadro delimitador alrededor de la cara de la imagen de entrada que Amazon Rekognition utilizó para la búsqueda.

Tipo: objeto [BoundingBox \(p. 740\)](#)

### [SearchedFaceConfidence \(p. 682\)](#)

Grado de confianza de que el `searchedFaceBoundingBox`, contiene un rostro.

Type: Float

Rango válido: Valor mínimo de 0. Valor máximo de 100.

## Errores

### [AccessDeniedException](#)

No tiene autorización para realizar la acción.

Código de estado HTTP: 400

### [ImageTooLargeException](#)

La imagen de entrada tamaño supera el límite permitido. Si estás llamando [DetectProtectiveEquipment \(p. 592\)](#), el tamaño de la imagen o la resolución supera el límite permitido. Para obtener más información, consulte [Directrices y cuotas de Amazon Rekognition \(p. 847\)](#).

Código de estado HTTP: 400

### [InternalServerError](#)

Amazon Lex ha tenido un problema de servicio. Pruebe la llamada de nuevo.

Código de estado HTTP: 500

### [InvalidImageFormatException](#)

No se admite el formato de imagen proporcionado.

Código de estado HTTP: 400

### [InvalidParameterException](#)

El parámetro de entrada infringió una restricción. Valide el parámetro antes de llamar a la operación de la API de nuevo.

Código de estado HTTP: 400

### [InvalidS3ObjectException](#)

Amazon Rekognition no puede obtener acceso al objeto de S3 especificado en la solicitud.

Código de estado HTTP: 400

### [ProvisionedThroughputExceededException](#)

El número de solicitudes ha superado su límite de rendimiento. Si necesita aumentar este límite, póngase en contacto con Amazon Rekognition.

Código de estado HTTP: 400

ResourceNotFoundException

No se puede encontrar el recurso especificado en la solicitud.

Código de estado HTTP: 400

ThrottlingException

Amazon Lex no puede procesar temporalmente la solicitud. Pruebe la llamada de nuevo.

Código de estado HTTP: 500

## Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- [AWS Command Line Interface](#)
- [SDK de AWS para .NET](#)
- [AWS SDK para C++](#)
- [AWS SDK para Go](#)
- [AWSSDK para Java V2](#)
- [AWS SDK para JavaScript](#)
- [SDK de AWS para PHP V3](#)
- [SDK de AWS para Python](#)
- [SDK de AWS para Ruby V3](#)

## StartCelebrityRecognition

Comienza el reconocimiento asíncrono de famosos en un vídeo almacenado.

Amazon Rekognition Video puede detectar famosos en un vídeo que debe almacenarse en un bucket de Amazon S3. Usar[Video \(p. 844\)](#)para especificar el nombre del depósito y el nombre de archivo del vídeo.[StartCelebrityRecognition](#)devuelve un identificador de trabajo ([JobId](#)) que utiliza para obtener los resultados del análisis. Cuando finaliza el análisis de reconocimiento de celebridades, Amazon Rekognition Video publica un estado de finalización en el tema Amazon Simple Notification Service que especifique en[NotificationChannel](#). Para obtener los resultados del análisis de reconocimiento de celebridades, primero compruebe que el valor de estado publicado en el tema de Amazon SNS es[SUCCEEDED](#). Si es así, llame[GetCelebrityRecognition \(p. 605\)](#)y pasa el identificador de trabajo ([JobId](#)) desde la llamada inicial hasta[StartCelebrityRecognition](#).

Para obtener más información, consulte [Reconocimiento de famosos \(p. 275\)](#).

### Sintaxis de la solicitud

```
{
 "ClientRequestToken": "string",
 "JobTag": "string",
 "NotificationChannel": {
 "RoleArn": "string",
 "SNSTopicArn": "string"
 },
 "Video": {
 "S3Object": {
 "Bucket": "string",
 "Name": "string",
 "Version": "string"
 }
 }
}
```

### Parámetros de solicitud

La solicitud acepta los siguientes datos en formato JSON.

#### [ClientRequestToken \(p. 685\)](#)

Token idempotente utilizado para identificar la solicitud de inicio. Si utilizas el mismo token con varios[StartCelebrityRecognition](#)solicitudes, lo mismo[JobId](#)se devuelve. Usar[ClientRequestToken](#)para evitar que el mismo trabajo se inicie accidentalmente más de una vez.

Type: Cadena

Restricciones de longitud: Longitud mínima de 1. La longitud máxima es 64.

Patrón: ^[a-zA-Z0-9-\_]+\$

Obligatorio: No

#### [JobTag \(p. 685\)](#)

Identificador que específicas que se devuelve en la notificación de finalización publicada en el tema de Amazon Simple Notification Service. Por ejemplo, puede utilizar[JobTag](#)para agrupar trabajos relacionados e identificarlos en la notificación de finalización.

Type: Cadena

Restricciones de longitud: Longitud mínima de 1. La longitud máxima es de 256 caracteres.

Patrón: [ a-zA-Z0-9\_.\-\: ]+

Obligatorio: No

[NotificationChannel \(p. 685\)](#)

El tema de Amazon SNS ARN en el que desea que Amazon Rekognition Video publique el estado de finalización del análisis de reconocimiento de celebridades. El tema de Amazon SNS debe tener un nombre de tema que comience por AmazonRekognitionsi utiliza la política de permisos AmazonRekognitionServiceRole.

Tipo: objeto [NotificationChannel \(p. 799\)](#)

Obligatorio: No

[Video \(p. 685\)](#)

El vídeo en el que quieres reconocer a las celebridades. El vídeo debe almacenarse en un bucket de Amazon S3.

Tipo: objeto [Video \(p. 844\)](#)

Obligatorio: Sí

## Sintaxis de la respuesta

```
{
 "JobId": "string"
}
```

## Elementos de respuesta

Si la acción se realiza correctamente, el servicio devuelve una respuesta HTTP 200.

El servicio devuelve los datos siguientes en formato JSON.

[JobId \(p. 686\)](#)

Identificador del trabajo de análisis de reconocimiento de celebridades. Usar JobId para identificar el trabajo en una llamada posterior a `GetCelebrityRecognition`.

Type: Cadena

Restricciones de longitud: Longitud mínima de 1. La longitud máxima es 64.

Patrón: ^[ a-zA-Z0-9-\_ ]+\$

## Errores

[AccessDeniedException](#)

No tiene autorización para realizar la acción.

Código de estado HTTP: 400

**IdempotentParameterMismatchException**

**UNAClientRequestToken**Se ha reutilizado con una operación, pero al menos uno de los demás parámetros de entrada es distinto de la llamada anterior a la operación.

Código de estado HTTP: 400

**InternalServerError**

Amazon Lex ha tenido un problema de servicio. Pruebe la llamada de nuevo.

Código de estado HTTP: 500

**InvalidParameterException**

El parámetro de entrada infringió una restricción. Valide el parámetro antes de llamar a la operación de la API de nuevo.

Código de estado HTTP: 400

**InvalidS3ObjectException**

Amazon Rekognition no puede obtener acceso al objeto de S3 especificado en la solicitud.

Código de estado HTTP: 400

**LimitExceededException**

Se ha superado un límite de servicio de Amazon Rekognition. Por ejemplo, si inicia demasiados trabajos de Amazon Rekognition Video simultáneamente, llama para iniciar operaciones (`StartLabelDetection`, por ejemplo) elevará un `LimitExceeded`Exception excepción (código de estado HTTP: 400) hasta que el número de trabajos ejecutados simultáneamente se encuentre por debajo del límite de servicio de Amazon Rekognition.

Código de estado HTTP: 400

**ProvisionedThroughputExceededException**

El número de solicitudes ha superado su límite de rendimiento. Si necesita aumentar este límite, póngase en contacto con Amazon Rekognition.

Código de estado HTTP: 400

**ThrottlingException**

Amazon Lex no puede procesar temporalmente la solicitud. Pruebe la llamada de nuevo.

Código de estado HTTP: 500

**VideoTooLargeException**

El tamaño del archivo o la duración del medio suministrado es demasiado grande. El tamaño de archivo máximo es de 10 GB. La duración máxima es de 6 horas.

Código de estado HTTP: 400

## Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- [AWS Command Line Interface](#)
- [SDK de AWS para .NET](#)
- [AWS SDK para C++](#)

- [AWS SDK para Go](#)
- [AWSSDK para Java V2](#)
- [AWS SDK para JavaScript](#)
- [SDK de AWS para PHP V3](#)
- [SDK de AWS para Python](#)
- [SDK de AWS para Ruby V3](#)

## StartContentModeration

Inicia la detección asíncrona de contenido inapropiado, no deseado u ofensivo en un vídeo. Para obtener una lista de etiquetas de moderación en Amazon Rekognition, consulte[Uso de las API de moderación de imagen y vídeo](#).

Amazon Rekognition Video puede moderar el contenido de un vídeo almacenado en un bucket de Amazon S3. Usar[Video \(p. 844\)](#)para especificar el nombre del depósito y el nombre de archivo del vídeo.`StartContentModeration`devuelve un identificador de trabajo (`JobId`) que utiliza para obtener los resultados del análisis. Cuando finaliza el análisis de contenido, Amazon Rekognition Video publica un estado de finalización en el tema Amazon Simple Notification Service que especifique en[NotificationChannel](#).

Para obtener los resultados del análisis de contenido, primero compruebe que el valor de estado publicado en el tema de Amazon SNS es[SUCCEEDED](#). Si es así, llame[GetContentModeration \(p. 611\)](#)y pasa el identificador de trabajo (`JobId`) desde la llamada inicial hasta[StartContentModeration](#).

Para obtener más información, consulte [Moderación de contenido \(p. 298\)](#).

## Sintaxis de la solicitud

```
{
 "ClientRequestToken": "string",
 "JobTag": "string",
 "MinConfidence": number,
 "NotificationChannel": {
 "RoleArn": "string",
 "SNSTopicArn": "string"
 },
 "Video": {
 "S3Object": {
 "Bucket": "string",
 "Name": "string",
 "Version": "string"
 }
 }
}
```

## Parámetros de solicitud

La solicitud acepta los siguientes datos en formato JSON.

### [ClientRequestToken \(p. 689\)](#)

Token idempotente utilizado para identificar la solicitud de inicio. Si utilizas el mismo token con varios[StartContentModeration](#)solicitudes, lo mismo[JobId](#)se devuelve. Usar[ClientRequestToken](#)para evitar que el mismo trabajo se inicie accidentalmente más de una vez.

Type: Cadena

Restricciones de longitud: Longitud mínima de 1. La longitud máxima es 64.

Patrón: ^[a-zA-Z0-9-\_]+\$

Obligatorio No

### [JobTag \(p. 689\)](#)

Identificador que especificas que se devuelve en la notificación de finalización publicada en el tema de Amazon Simple Notification Service. Por ejemplo, puede utilizar `JobTag` para agrupar trabajos relacionados e identificarlos en la notificación de finalización.

Type: Cadena

Restricciones de longitud: Longitud mínima de 1. La longitud máxima es de 256 caracteres.

Patrón: [ a-zA-Z0-9\_.\-\: ]+

Obligatorio No

### [MinConfidence \(p. 689\)](#)

Especifica la confianza mínima que debe tener Amazon Rekognition para devolver una etiqueta de contenido moderada. La confianza representa cuán determinado Amazon Rekognition se identifica correctamente el contenido moderado. 0 es la confianza más baja. 100 es la confianza más alta. Amazon Rekognition no devuelve etiquetas de contenido moderado con un nivel de confianza inferior al valor especificado. Si no especifica `MinConfidence`, `GetContentModeration` devuelve etiquetas con valores de confianza superiores o iguales al 50 por ciento.

Type: Float

Rango válido Valor mínimo de 0. Valor máximo de 100.

Obligatorio No

### [NotificationChannel \(p. 689\)](#)

ARN del tema de Amazon SNS en el que desea que Amazon Rekognition Video publique el estado de finalización del análisis de contenido. El tema de Amazon SNS debe tener un nombre de tema que comience por `AmazonRekognition` si utilizas la política de permisos de `AmazonRekognitionServiceRole` para acceder al tema.

Tipo: objeto [NotificationChannel \(p. 799\)](#)

Obligatorio No

### [Video \(p. 689\)](#)

Vídeo en el que quieres detectar contenido inapropiado, no deseado u ofensivo. El vídeo debe almacenarse en un bucket de Amazon S3.

Tipo: objeto [Video \(p. 844\)](#)

Obligatorio Sí

## Sintaxis de la respuesta

```
{
 "JobId": "string"
}
```

## Elementos de respuesta

Si la acción se realiza correctamente, el servicio devuelve una respuesta HTTP 200.

El servicio devuelve los datos siguientes en formato JSON.

### [JobId \(p. 690\)](#)

El identificador para el trabajo de análisis de contenido. Usar `JobId` para identificar el trabajo en una llamada posterior a `GetContentModeration`.

Type: Cadena

Restricciones de longitud: Longitud mínima de 1. La longitud máxima es 64.

Patrón: ^[a-zA-Z0-9-\_]+\$

## Errores

### AccessDeniedException

No tiene autorización para realizar la acción.

Código de estado HTTP: 400

### IdempotentParameterMismatchException

`UNAClientRequestToken`El parámetro de entrada se ha reutilizado con una operación, pero al menos uno de los demás parámetros de entrada es distinto de la llamada anterior a la operación.

Código de estado HTTP: 400

### InternalServerError

Amazon Lex ha tenido un problema de servicio. Pruebe la llamada de nuevo.

Código de estado HTTP: 500

### InvalidParameterException

El parámetro de entrada infringió una restricción. Valide el parámetro antes de llamar a la operación de la API de nuevo.

Código de estado HTTP: 400

### InvalidS3ObjectException

Amazon Rekognition no puede obtener acceso al objeto de S3 especificado en la solicitud.

Código de estado HTTP: 400

### LimitExceededException

Se ha superado un límite de servicio de Amazon Rekognition. Por ejemplo, si inicia demasiados trabajos de Amazon Rekognition Video simultáneamente, llama para iniciar operaciones (`StartLabelDetection`, por ejemplo) elevará un `LimitExceededException` excepción (código de estado HTTP: 400) hasta que el número de trabajos ejecutados simultáneamente se encuentre por debajo del límite de servicio de Amazon Rekognition.

Código de estado HTTP: 400

### ProvisionedThroughputExceededException

El número de solicitudes ha superado su límite de rendimiento. Si necesita aumentar este límite, póngase en contacto con Amazon Rekognition.

Código de estado HTTP: 400

### ThrottlingException

Amazon Lex no puede procesar temporalmente la solicitud. Pruebe la llamada de nuevo.

Código de estado HTTP: 500

**VideoTooLargeException**

El tamaño del archivo o la duración del medio suministrado es demasiado grande. El tamaño de archivo máximo es de 10 GB. La duración máxima es de 6 horas.

Código de estado HTTP: 400

## Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- [AWS Command Line Interface](#)
- [SDK de AWS para .NET](#)
- [AWS SDK para C++](#)
- [AWS SDK para Go](#)
- [AWSSDK para Java V2](#)
- [AWS SDK para JavaScript](#)
- [SDK de AWS para PHP V3](#)
- [SDK de AWS para Python](#)
- [SDK de AWS para Ruby V3](#)

## StartFaceDetection

Comienza la detección asincrónica de rostros en un vídeo almacenado.

Amazon Rekognition Video puede detectar rostros en un vídeo almacenado en un bucket de Amazon S3. Usar[Video \(p. 844\)](#)para especificar el nombre del depósito y el nombre de archivo del vídeo.[StartFaceDetection](#)devuelve un identificador de trabajo (`JobId`) que utiliza para obtener los resultados de la operación. Cuando finaliza la detección de rostros, Amazon Rekognition Video publica un estado de finalización en el tema Amazon Simple Notification Service que especifique `enNotificationChannel1`. Para obtener los resultados de la operación de detección de rostros, compruebe primero que el valor de estado publicado en el tema de Amazon SNS es `SUCCEEDED`. Si es así, llame[GetFaceDetection \(p. 615\)](#)y pasa el identificador de trabajo (`JobId`) desde la llamada inicial hasta[StartFaceDetection](#).

Para obtener más información, consulte [Detección de rostros en un vídeo almacenado \(p. 172\)](#).

### Sintaxis de la solicitud

```
{
 "ClientRequestToken": "string",
 "FaceAttributes": "string",
 "JobTag": "string",
 "NotificationChannel": {
 "RoleArn": "string",
 "SNSTopicArn": "string"
 },
 "Video": {
 "S3Object": {
 "Bucket": "string",
 "Name": "string",
 "Version": "string"
 }
 }
}
```

### Parámetros de solicitud

La solicitud acepta los siguientes datos en formato JSON.

#### [ClientRequestToken \(p. 693\)](#)

Token idempotente utilizado para identificar la solicitud de inicio. Si utilizas el mismo token con varios[StartFaceDetection](#)solicitudes, lo mismo[JobId](#)se devuelve. Usar[ClientRequestToken](#)para evitar que el mismo trabajo se inicie accidentalmente más de una vez.

Type: Cadena

Restricciones de longitud: Longitud mínima de 1. La longitud máxima es 64.

Patrón: ^[a-zA-Z0-9-\_]+\$

Obligatorio: No

#### [FaceAttributes \(p. 693\)](#)

Los atributos de cara que desea que se devuelvan.

**DEFAULT**- Se devuelve el siguiente subconjunto de atributos faciales: BoundingBox, confianza, postura, calidad y puntos de referencia.

**ALL**- Se devuelven todos los atributos faciales.

Type: Cadena

Valores válidos: `DEFAULT` | `ALL`

Obligatorio: No

[JobTag \(p. 693\)](#)

Identificador que especificas que se devuelve en la notificación de finalización publicada en el tema de Amazon Simple Notification Service. Por ejemplo, puede utilizar `JobTag` para agrupar trabajos relacionados e identificarlos en la notificación de finalización.

Type: Cadena

Restricciones de longitud: Longitud mínima de 1. La longitud máxima es de 256 caracteres.

Patrón: `[a-zA-Z0-9_.\-\:]+`

Obligatorio: No

[NotificationChannel \(p. 693\)](#)

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 rostros. El tema de Amazon SNS debe tener un nombre de tema que comience por `AmazonRekognition` si se utiliza la política de permisos `AmazonRekognitionServiceRole`.

Tipo: objeto [NotificationChannel \(p. 799\)](#)

Obligatorio: No

[Video \(p. 693\)](#)

El vídeo en el que quieres detectar rostros. El vídeo debe almacenarse en un bucket de Amazon S3.

Tipo: objeto [Video \(p. 844\)](#)

Obligatorio: Sí

## Sintaxis de la respuesta

```
{
 "JobId": "string"
}
```

## Elementos de respuesta

Si la acción se realiza correctamente, el servicio devuelve una respuesta HTTP 200.

El servicio devuelve los datos siguientes en formato JSON.

[JobId \(p. 694\)](#)

Identificador del trabajo de detección de rostros. Usar `JobId` para identificar el trabajo en una llamada posterior a `GetFaceDetection`.

Type: Cadena

Restricciones de longitud: Longitud mínima de 1. La longitud máxima es 64.

Patrón: ^[ a-zA-Z0-9-\_ ]+\$

## Errores

### AccessDeniedException

No tiene autorización para realizar la acción.

Código de estado HTTP: 400

### IdempotentParameterMismatchException

**UNAclientRequestToken**El parámetro de entrada se ha reutilizado con una operación, pero al menos uno de los demás parámetros de entrada es distinto de la llamada anterior a la operación.

Código de estado HTTP: 400

### InternalServerError

Amazon Lex ha tenido un problema de servicio. Pruebe la llamada de nuevo.

Código de estado HTTP: 500

### InvalidParameterException

El parámetro de entrada infringió una restricción. Valide el parámetro antes de llamar a la operación de la API de nuevo.

Código de estado HTTP: 400

### InvalidS3ObjectException

Amazon Rekognition no puede obtener acceso al objeto de S3 especificado en la solicitud.

Código de estado HTTP: 400

### LimitExceededException

Se ha superado un límite de servicio de Amazon Rekognition. Por ejemplo, si inicia demasiados trabajos de Amazon Rekognition Video simultáneamente, llama para iniciar operaciones (`StartLabelDetection`, por ejemplo) elevará un `LimitExceededException` excepción (código de estado HTTP: 400) hasta que el número de trabajos ejecutados simultáneamente se encuentre por debajo del límite de servicio de Amazon Rekognition.

Código de estado HTTP: 400

### ProvisionedThroughputExceededException

El número de solicitudes ha superado su límite de rendimiento. Si necesita aumentar este límite, póngase en contacto con Amazon Rekognition.

Código de estado HTTP: 400

### ThrottlingException

Amazon Lex no puede procesar temporalmente la solicitud. Pruebe la llamada de nuevo.

Código de estado HTTP: 500

### VideoTooLargeException

El tamaño del archivo o la duración del medio suministrado es demasiado grande. El tamaño de archivo máximo es de 10 GB. La duración máxima es de 6 horas.

Código de estado HTTP: 400

## Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- [AWS Command Line Interface](#)
- [SDK de AWS para .NET](#)
- [AWS SDK para C++](#)
- [AWS SDK para Go](#)
- [AWSSDK para Java V2](#)
- [AWS SDK para JavaScript](#)
- [SDK de AWS para PHP V3](#)
- [SDK de AWS para Python](#)
- [SDK de AWS para Ruby V3](#)

## StartFaceSearch

Comienza la búsqueda asíncrona de rostros en una colección que coincidan con los rostros de las personas detectadas en un vídeo.

El vídeo debe almacenarse en un bucket de Amazon S3. Usar[Video \(p. 844\)](#)para especificar el nombre del depósito y el nombre de archivo del vídeo.[StartFaceSearch](#)devuelve un identificador de trabajo ([JobId](#)) que utiliza para obtener los resultados de la búsqueda una vez finalizada la búsqueda. Cuando finaliza la búsqueda, Amazon Rekognition Video publica un estado de finalización en el tema Amazon Simple Notification Service que especifique [enNotificationChannel](#). Para obtener los resultados de la búsqueda, primero compruebe que el valor de estado publicado en el tema de Amazon SNS es [SUCCEEDED](#). Si es así, llame[GetFaceSearch \(p. 620\)](#)y pasa el identificador de trabajo ([JobId](#)) desde la llamada inicial hasta[startFaceSearch](#). Para obtener más información, consulte [Búsqueda de rostros en vídeos almacenados \(p. 234\)](#).

## Sintaxis de la solicitud

```
{
 "ClientRequestToken": "string",
 "CollectionId": "string",
 "FaceMatchThreshold": number,
 "JobTag": "string",
 "NotificationChannel": {
 "RoleArn": "string",
 "SNSTopicArn": "string"
 },
 "Video": {
 "S3Object": {
 "Bucket": "string",
 "Name": "string",
 "Version": "string"
 }
 }
}
```

## Parámetros de solicitud

La solicitud acepta los siguientes datos en formato JSON.

### [ClientRequestToken \(p. 697\)](#)

Token idempotente utilizado para identificar la solicitud de inicio. Si utilizas el mismo token con varios[StartFaceSearch](#)solicitudes, lo mismo[JobId](#)se devuelve. Usar[ClientRequestToken](#)para evitar que el mismo trabajo se inicie accidentalmente más de una vez.

Type: Cadena

Restricciones de longitud: Longitud mínima de 1. La longitud máxima es 64.

Patrón: ^[a-zA-Z0-9-\_]+\$

Obligatorio: No

### [CollectionId \(p. 697\)](#)

Identificador de la colección que contiene los rostros que desea buscar.

Type: Cadena

Restricciones de longitud: Longitud mínima de 1. La longitud máxima es de 255 caracteres.

Patrón: [ a-zA-Z0-9\_.\-\: ]+

Obligatorio: Sí

[FaceMatchThreshold \(p. 697\)](#)

La confianza mínima en la persona coincide con la devolución. Por ejemplo, no devuelvas ningún partido en el que la confianza en los partidos sea inferior al 70%. El valor predeterminado es el 80%.

Type: Float

Rango válido: Valor mínimo de 0. Valor máximo de 100.

Obligatorio: No

[JobTag \(p. 697\)](#)

Identificador que especificas que se devuelve en la notificación de finalización publicada en el tema de Amazon Simple Notification Service. Por ejemplo, puede utilizar JobTag para agrupar trabajos relacionados e identificarlos en la notificación de finalización.

Type: Cadena

Restricciones de longitud: Longitud mínima de 1. La longitud máxima es de 256 caracteres.

Patrón: [ a-zA-Z0-9\_.\-\: ]+

Obligatorio: No

[NotificationChannel \(p. 697\)](#)

El ARN del tema de Amazon SNS en el que desea que Amazon Rekognition Video publique el estado de finalización de la búsqueda. El tema de Amazon SNS debe tener un nombre de tema que comience por AmazonRekognition si utilizas la política de permisos de AmazonRekognitionServiceRole para acceder al tema.

Tipo: objeto [NotificationChannel \(p. 799\)](#)

Obligatorio: No

[Video \(p. 697\)](#)

El vídeo que desea buscar. El vídeo debe almacenarse en un bucket de Amazon S3.

Tipo: objeto [Video \(p. 844\)](#)

Obligatorio: Sí

## Sintaxis de la respuesta

```
{
 "JobId": "string"
}
```

## Elementos de respuesta

Si la acción se realiza correctamente, el servicio devuelve una respuesta HTTP 200.

El servicio devuelve los datos siguientes en formato JSON.

### [JobId \(p. 698\)](#)

Identificador del trabajo de búsqueda. Usar `JobId` para identificar el trabajo en una llamada posterior a `GetFaceSearch`.

Type: Cadena

Restricciones de longitud: Longitud mínima de 1. La longitud máxima es 64.

Patrón: ^[a-zA-Z0-9-\_]+\$

## Errores

### AccessDeniedException

No tiene autorización para realizar la acción.

Código de estado HTTP: 400

### IdempotentParameterMismatchException

`UNAClientRequestToken`El parámetro de entrada se ha reutilizado con una operación, pero al menos uno de los demás parámetros de entrada es distinto de la llamada anterior a la operación.

Código de estado HTTP: 400

### InternalServerError

Amazon Lex ha tenido un problema de servicio. Pruebe la llamada de nuevo.

Código de estado HTTP: 500

### InvalidParameterException

El parámetro de entrada infringió una restricción. Valide el parámetro antes de llamar a la operación de la API de nuevo.

Código de estado HTTP: 400

### InvalidS3ObjectException

Amazon Rekognition no puede obtener acceso al objeto de S3 especificado en la solicitud.

Código de estado HTTP: 400

### LimitExceededException

Se ha superado un límite de servicio de Amazon Rekognition. Por ejemplo, si inicia demasiados trabajos de Amazon Rekognition Video simultáneamente, llama para iniciar operaciones (`StartLabelDetection`, por ejemplo) elevará un `LimitExceededException` excepción (código de estado HTTP: 400) hasta que el número de trabajos ejecutados simultáneamente se encuentre por debajo del límite de servicio de Amazon Rekognition.

Código de estado HTTP: 400

### ProvisionedThroughputExceededException

El número de solicitudes ha superado su límite de rendimiento. Si necesita aumentar este límite, póngase en contacto con Amazon Rekognition.

Código de estado HTTP: 400

### ResourceNotFoundException

No se puede encontrar el recurso especificado en la solicitud.

Código de estado HTTP: 400  
**ThrottlingException**

Amazon Lex no puede procesar temporalmente la solicitud. Pruebe la llamada de nuevo.

Código de estado HTTP: 500  
**VideoTooLargeException**

El tamaño del archivo o la duración del medio suministrado es demasiado grande. El tamaño de archivo máximo es de 10 GB. La duración máxima es de 6 horas.

Código de estado HTTP: 400

## Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- [AWS Command Line Interface](#)
- [SDK de AWS para .NET](#)
- [AWS SDK para C++](#)
- [AWS SDK para Go](#)
- [AWSSDK para Java V2](#)
- [AWS SDK para JavaScript](#)
- [SDK de AWS para PHP V3](#)
- [SDK de AWS para Python](#)
- [SDK de AWS para Ruby V3](#)

## StartLabelDetection

Comienza la detección asincrónica de etiquetas en un vídeo almacenado.

Amazon Rekognition Video puede detectar etiquetas en un vídeo. Las etiquetas son instancias de entidades del mundo real. Esto incluye objetos como flores, árboles y mesa; eventos como bodas, graduaciones y fiestas de cumpleaños; conceptos como paisaje, noche y naturaleza; y actividades como una persona que sale de un automóvil o una persona que esquia.

El vídeo debe almacenarse en un bucket de Amazon S3. Usar[Video \(p. 844\)](#)para especificar el nombre del depósito y el nombre de archivo del vídeo.[StartLabelDetection](#)devuelve un identificador de trabajo ([JobId](#)) que utiliza para obtener los resultados de la operación. Cuando finaliza la detección de etiquetas, Amazon Rekognition Video publica un estado de finalización en el tema Amazon Simple Notification Service que especifique en[NotificationChannel](#).

Para obtener los resultados de la operación de detección de etiquetas, compruebe primero que el valor de estado publicado en el tema de Amazon SNS es[SUCCEEDED](#). Si es así, llame[GetLabelDetection \(p. 626\)](#)y pasa el identificador de trabajo ([JobId](#)) desde la llamada inicial hasta[StartLabelDetection](#).

## Sintaxis de la solicitud

```
{
 "ClientRequestToken": "string",
 "JobTag": "string",
 "MinConfidence": number,
 "NotificationChannel": {
 "RoleArn": "string",
 "SNSTopicArn": "string"
 },
 "Video": {
 "S3Object": {
 "Bucket": "string",
 "Name": "string",
 "Version": "string"
 }
 }
}
```

## Parámetros de solicitud

La solicitud acepta los siguientes datos en formato JSON.

### [ClientRequestToken \(p. 701\)](#)

Token idempotente utilizado para identificar la solicitud de inicio. Si utilizas el mismo token con varios[StartLabelDetection](#)solicitudes, lo mismo[JobId](#)se devuelve. Usar[ClientRequestToken](#)para evitar que el mismo trabajo se inicie accidentalmente más de una vez.

Type: Cadena

Restricciones de longitud: Longitud mínima de 1. La longitud máxima es 64.

Patrón: ^[a-zA-Z0-9-\_]+\$

Obligatorio No

### [JobTag \(p. 701\)](#)

Identificador que especificas que se devuelve en la notificación de finalización publicada en el tema de Amazon Simple Notification Service. Por ejemplo, puede utilizar `JobTag` para agrupar trabajos relacionados e identificarlos en la notificación de finalización.

Type: Cadena

Restricciones de longitud: Longitud mínima de 1. La longitud máxima es de 256 caracteres.

Patrón: [ a-zA-Z0-9\_.\-\:]<sup>+</sup>

Obligatorio No

### [MinConfidence \(p. 701\)](#)

Especifica la confianza mínima que debe tener Amazon Rekognition Video para devolver una etiqueta detectada. La confianza representa cuán cierto Amazon Rekognition es que una etiqueta está correctamente identificada. 0 es la confianza más baja. 100 es la confianza más alta. Amazon Rekognition Video no devuelve ninguna etiqueta con un nivel de confianza inferior al valor especificado.

Si no especifica `MinConfidence`, la operación devuelve etiquetas con valores de confianza superiores o iguales al 50 por ciento.

Type: Float

Rango válido: Valor mínimo de 0. Valor máximo de 100.

Obligatorio No

### [NotificationChannel \(p. 701\)](#)

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. El tema de Amazon SNS debe tener un nombre de tema que comience por `AmazonRekognition` si se utiliza la política de permisos `AmazonRekognitionServiceRole`.

Tipo: objeto [NotificationChannel \(p. 799\)](#)

Obligatorio No

### [Video \(p. 701\)](#)

Vídeo en el que quieres detectar etiquetas. El vídeo debe almacenarse en un bucket de Amazon S3.

Tipo: objeto [Video \(p. 844\)](#)

Obligatorio Sí

## Sintaxis de la respuesta

```
{
 "JobId": "string"
}
```

## Elementos de respuesta

Si la acción se realiza correctamente, el servicio devuelve una respuesta HTTP 200.

El servicio devuelve los datos siguientes en formato JSON.

### [JobId \(p. 702\)](#)

Identificador del trabajo de detección de etiquetas. Usar `JobId` para identificar el trabajo en una llamada posterior a `GetLabelDetection`.

Type: Cadena

Restricciones de longitud: Longitud mínima de 1. La longitud máxima es 64.

Patrón: ^[a-zA-Z0-9-\_]+\$

## Errores

### AccessDeniedException

No tiene autorización para realizar la acción.

Código de estado HTTP: 400

### IdempotentParameterMismatchException

`UNAClientRequestToken`Se ha reutilizado un parámetro de entrada con una operación, pero al menos uno de los demás parámetros de entrada es distinto de la llamada anterior a la operación.

Código de estado HTTP: 400

### InternalServerError

Amazon Lex ha tenido un problema de servicio. Pruebe la llamada de nuevo.

Código de estado HTTP: 500

### InvalidParameterException

El parámetro de entrada infringió una restricción. Valide el parámetro antes de llamar a la operación de la API de nuevo.

Código de estado HTTP: 400

### InvalidS3ObjectException

Amazon Rekognition no puede obtener acceso al objeto de S3 especificado en la solicitud.

Código de estado HTTP: 400

### LimitExceededException

Se ha superado un límite de servicio de Amazon Rekognition. Por ejemplo, si inicia demasiados trabajos de Amazon Rekognition Video simultáneamente, llama para iniciar operaciones (`StartLabelDetection`, por ejemplo) elevará un `LimitExceededException`excepción (código de estado HTTP: 400) hasta que el número de trabajos ejecutados simultáneamente se encuentre por debajo del límite de servicio de Amazon Rekognition.

Código de estado HTTP: 400

### ProvisionedThroughputExceededException

El número de solicitudes ha superado su límite de rendimiento. Si necesita aumentar este límite, póngase en contacto con Amazon Rekognition.

Código de estado HTTP: 400

### ThrottlingException

Amazon Lex no puede procesar temporalmente la solicitud. Pruebe la llamada de nuevo.

Código de estado HTTP: 500

**VideoTooLargeException**

El tamaño del archivo o la duración del medio suministrado es demasiado grande. El tamaño de archivo máximo es de 10 GB. La duración máxima es de 6 horas.

Código de estado HTTP: 400

## Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- [AWS Command Line Interface](#)
- [SDK de AWS para .NET](#)
- [AWS SDK para C++](#)
- [AWS SDK para Go](#)
- [AWSSDK para Java V2](#)
- [AWS SDK para JavaScript](#)
- [SDK de AWS para PHP V3](#)
- [SDK de AWS para Python](#)
- [SDK de AWS para Ruby V3](#)

## StartPersonTracking

Comienza el seguimiento asincrónico de la ruta de una persona en un vídeo almacenado.

Amazon Rekognition Video puede realizar un seguimiento del recorrido de las personas en un vídeo almacenado en un bucket de Amazon S3. Usar[Video \(p. 844\)](#)para especificar el nombre del depósito y el nombre de archivo del vídeo.[StartPersonTracking](#)devuelve un identificador de trabajo ([JobId](#)) que utiliza para obtener los resultados de la operación. Cuando finaliza la detección de etiquetas, Amazon Rekognition publica un estado de finalización en el tema Amazon Simple Notification Service que especifique en[NotificationChannel](#).

Para obtener los resultados de la operación de detección de personas, compruebe primero que el valor de estado publicado en el tema de Amazon SNS es[SUCCEEDED](#). Si es así, llame[GetPersonTracking \(p. 630\)](#)y pasa el identificador de trabajo ([JobId](#)) desde la llamada inicial hasta[StartPersonTracking](#).

### Sintaxis de la solicitud

```
{
 "ClientRequestToken": "string",
 "JobTag": "string",
 "NotificationChannel": {
 "RoleArn": "string",
 "SNSTopicArn": "string"
 },
 "Video": {
 "S3Object": {
 "Bucket": "string",
 "Name": "string",
 "Version": "string"
 }
 }
}
```

### Parámetros de solicitud

La solicitud acepta los siguientes datos en formato JSON.

#### [ClientRequestToken \(p. 705\)](#)

Token idempotente utilizado para identificar la solicitud de inicio. Si utilizas el mismo token con varios[StartPersonTracking](#)solicitudes, lo mismo[JobId](#)se devuelve. Usar[ClientRequestToken](#)para evitar que el mismo trabajo se inicie accidentalmente más de una vez.

Type: Cadena

Restricciones de longitud: Longitud mínima de 1. La longitud máxima es 64.

Patrón: ^[a-zA-Z0-9-\_]+\$

Obligatorio: No

#### [JobTag \(p. 705\)](#)

Identificador que especificas que se devuelve en la notificación de finalización publicada en el tema de Amazon Simple Notification Service. Por ejemplo, puede utilizar[JobTag](#)para agrupar trabajos relacionados e identificarlos en la notificación de finalización.

Type: Cadena

Restricciones de longitud: Longitud mínima de 1. La longitud máxima es de 256 caracteres.

Patrón: [ a-zA-Z0-9\_.\-\:\ ]+

Obligatorio: No

[NotificationChannel \(p. 705\)](#)

El tema de Amazon SNS ARN en el que desea que Amazon Rekognition Video publique el estado de finalización de la operación de detección de personas. El tema de Amazon SNS debe tener un nombre de tema que comience por AmazonRekognitionServiceRole.

Tipo: objeto [NotificationChannel \(p. 799\)](#)

Obligatorio: No

[Video \(p. 705\)](#)

El vídeo en el que quieres detectar personas. El vídeo debe almacenarse en un bucket de Amazon S3.

Tipo: objeto [Video \(p. 844\)](#)

Obligatorio: Sí

## Sintaxis de la respuesta

```
{
 "JobId": "string"
}
```

## Elementos de respuesta

Si la acción se realiza correctamente, el servicio devuelve una respuesta HTTP 200.

El servicio devuelve los datos siguientes en formato JSON.

[JobId \(p. 706\)](#)

Identificador del trabajo de detección de personas. Usar JobId para identificar el trabajo en una llamada posterior a `GetPersonTracking`.

Type: Cadena

Restricciones de longitud: Longitud mínima de 1. La longitud máxima es 64.

Patrón: ^[ a-zA-Z0-9-\_ ]+\$

## Errores

[AccessDeniedException](#)

No tiene autorización para realizar la acción.

Código de estado HTTP: 400

[IdempotentParameterMismatchException](#)

[UNAClientRequestToken](#)Se ha reutilizado un parámetro de entrada con una operación, pero al menos uno de los demás parámetros de entrada es distinto de la llamada anterior a la operación.

Código de estado HTTP: 400

InternalServerError

Amazon Lex ha tenido un problema de servicio. Pruebe la llamada de nuevo.

Código de estado HTTP: 500

InvalidParameterException

El parámetro de entrada infringió una restricción. Valide el parámetro antes de llamar a la operación de la API de nuevo.

Código de estado HTTP: 400

InvalidS3ObjectException

Amazon Rekognition no puede obtener acceso al objeto de S3 especificado en la solicitud.

Código de estado HTTP: 400

LimitExceedededException

Se ha superado un límite de servicio de Amazon Rekognition. Por ejemplo, si inicia demasiados trabajos de Amazon Rekognition Video simultáneamente, llama para iniciar operaciones (`StartLabelDetection`, por ejemplo) elevará un `LimitExceedededException` excepción (código de estado HTTP: 400) hasta que el número de trabajos ejecutados simultáneamente se encuentre por debajo del límite de servicio de Amazon Rekognition.

Código de estado HTTP: 400

ProvisionedThroughputExceededException

El número de solicitudes ha superado su límite de rendimiento. Si necesita aumentar este límite, póngase en contacto con Amazon Rekognition.

Código de estado HTTP: 400

ThrottlingException

Amazon Lex no puede procesar temporalmente la solicitud. Pruebe la llamada de nuevo.

Código de estado HTTP: 500

VideoTooLargeException

El tamaño del archivo o la duración del medio suministrado es demasiado grande. El tamaño de archivo máximo es de 10 GB. La duración máxima es de 6 horas.

Código de estado HTTP: 400

## Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- [AWS Command Line Interface](#)
- [SDK de AWS para .NET](#)
- [AWS SDK para C++](#)
- [AWS SDK para Go](#)
- [AWSSDK para Java V2](#)
- [AWS SDK para JavaScript](#)

- [SDK de AWS para PHP V3](#)
- [SDK de AWS para Python](#)
- [SDK de AWS para Ruby V3](#)

## StartProjectVersion

Inicia la ejecución de la versión de un modelo. El inicio de un modelo tarda un tiempo en completarse. Para comprobar el estado actual del modelo, utilice[DescribeProjectVersions \(p. 564\)](#).

Una vez que el modelo se esté ejecutando, puedes detectar etiquetas personalizadas en nuevas imágenes llamando[DetectCustomLabels \(p. 573\)](#).

### Note

Se le cobrará por la cantidad de tiempo durante la ejecución del modelo. Para detener un modelo en ejecución, llame a[StopProjectVersion \(p. 722\)](#).

Esta operación requiere permisos para realizar la acción `rekognition:StartProjectVersion`.

## Sintaxis de la solicitud

```
{
 "MinInferenceUnits": number,
 "ProjectVersionArn": "string"
}
```

## Parámetros de solicitud

La solicitud acepta los siguientes datos en formato JSON.

### [MinInferenceUnits \(p. 709\)](#)

El número mínimo de unidades de inferencia que se van a utilizar. Una única unidad de inferencia representa 1 hora de procesamiento y puede admitir hasta 5 transacciones por segundo (TPS). Utilice un número mayor para aumentar el rendimiento TPS de su modelo. Se le cobrará por el número de unidades de inferencia que utilice.

Type: Entero

Rango válido: Valor mínimo de 1.

Obligatorio: Sí

### [ProjectVersionArn \(p. 709\)](#)

El nombre de recurso de Amazon (ARN) de la versión del modelo que desea iniciar.

Type: Cadena

Restricciones de longitud: Longitud mínima de 20. La longitud máxima es de 2048 caracteres.

Patrón: (^arn:[a-z\d-]+:rekognition:[a-z\d-]+\d{12}:project\/[a-zA-Z0-9\_.\-\\_]{1,255}\version\/[a-zA-Z0-9\_.\-\\_]{1,255}\/[0-9]+\$)

Obligatorio: Sí

## Sintaxis de la respuesta

```
{
 "Status": "string"
}
```

## Elementos de respuesta

Si la acción se realiza correctamente, el servicio devuelve una respuesta HTTP 200.

El servicio devuelve los datos siguientes en formato JSON.

### [Status \(p. 709\)](#)

El estado actual de ejecución del modelo.

Type: Cadena

Valores válidos: TRAINING\_IN\_PROGRESS | TRAINING\_COMPLETED | TRAINING\_FAILED | STARTING | RUNNING | FAILED | STOPPING | STOPPED | DELETING

## Errores

### AccessDeniedException

No tiene autorización para realizar la acción.

Código de estado HTTP: 400

### InternalServerError

Amazon Lex ha tenido un problema de servicio. Pruebe la llamada de nuevo.

Código de estado HTTP: 500

### InvalidArgumentException

El parámetro de entrada infringió una restricción. Valide el parámetro antes de llamar a la operación de la API de nuevo.

Código de estado HTTP: 400

### LimitExceededException

Se ha superado un límite de servicio de Amazon Rekognition. Por ejemplo, si inicia demasiados trabajos de Amazon Rekognition Video simultáneamente, llama para iniciar operaciones (`StartLabelDetection`, por ejemplo) elevará un `LimitExceededException` excepción (código de estado HTTP: 400) hasta que el número de trabajos ejecutados simultáneamente se encuentre por debajo del límite de servicio de Amazon Rekognition.

Código de estado HTTP: 400

### ProvisionedThroughputExceededException

El número de solicitudes ha superado su límite de rendimiento. Si necesita aumentar este límite, póngase en contacto con Amazon Rekognition.

Código de estado HTTP: 400

### ResourceInUseException

El recurso especificado ya se está utilizando.

Código de estado HTTP: 400

### ResourceNotFoundException

No se encuentra el recurso especificado en la solicitud.

Código de estado HTTP: 400

#### ThrottlingException

Amazon Lex no puede procesar temporalmente la solicitud. Pruebe la llamada de nuevo.

Código de estado HTTP: 500

### Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- [AWS Command Line Interface](#)
- [SDK de AWS para .NET](#)
- [AWS SDK para C++](#)
- [AWS SDK para Go](#)
- [AWSSDK para Java V2](#)
- [AWS SDK para JavaScript](#)
- [SDK de AWS para PHP V3](#)
- [SDK de AWS para Python](#)
- [SDK de AWS para Ruby V3](#)

## StartSegmentDetection

Comienza la detección asincrónica de la detección de segmentos en un vídeo almacenado.

Amazon Rekognition Video puede detectar segmentos en un vídeo almacenado en un bucket de Amazon S3. Usar[Video \(p. 844\)](#)para especificar el nombre del depósito y el nombre de archivo del vídeo.[StartSegmentDetection](#)devuelve un identificador de trabajo ([JobId](#)) que utiliza para obtener los resultados de la operación. Cuando finaliza la detección de segmentos, Amazon Rekognition Video publica un estado de finalización en el tema Amazon Simple Notification Service que especifique en[NotificationChannel](#).

Puede utilizar el parámetro de entrada [Filters \(StartSegmentDetectionFilters \(p. 824\)\)](#) para especificar la confianza de detección mínima que se devuelve en la respuesta. En [Filters](#), use [ShotFilter \(StartShotDetectionFilter \(p. 825\)\)](#) para filtrar las tomas detectadas. [UsarTechnicalCueFilter\(StartTechnicalCueDetectionFilter \(p. 826\)\)](#) para filtrar las señales técnicas.

Para obtener los resultados de la operación de detección de segmentos, compruebe primero que el valor de estado publicado en el tema de Amazon SNS es [SUCCEEDED](#). si es así, llame[GetSegmentDetection \(p. 635\)](#)y pasa el identificador de trabajo ([JobId](#)) desde la llamada inicial hasta[StartSegmentDetection](#).

Para obtener más información, consulte [Detección de segmentos de vídeo en vídeo almacenado \(p. 338\)](#).

## Sintaxis de la solicitud

```
{
 "ClientRequestToken": "string",
 "Filters": {
 "ShotFilter": {
 "MinSegmentConfidence": number
 },
 "TechnicalCueFilter": {
 "BlackFrame": {
 "MaxPixelThreshold": number,
 "MinCoveragePercentage": number
 },
 "MinSegmentConfidence": number
 }
 },
 "JobTag": "string",
 "NotificationChannel": {
 "RoleArn": "string",
 "SNSTopicArn": "string"
 },
 "SegmentTypes": ["string"],
 "Video": {
 "S3Object": {
 "Bucket": "string",
 "Name": "string",
 "Version": "string"
 }
 }
}
```

## Parámetros de solicitud

La solicitud acepta los siguientes datos en formato JSON.

### [ClientRequestToken \(p. 712\)](#)

Token idempotente utilizado para identificar la solicitud de inicio. Si utilizas el mismo token con varios `StartSegmentDetection` solicitudes, lo mismo `JobId` se devuelve. Usa `ClientRequestToken` para evitar que el mismo trabajo se inicie accidentalmente más de una vez.

Type: Cadena

Restricciones de longitud: Longitud mínima de 1. La longitud máxima es 64.

Patrón: ^[a-zA-Z0-9-\_]+\$

Obligatorio: No

### [Filters \(p. 712\)](#)

Filtros para detección técnica de tacos o disparos.

Tipo: objeto [StartSegmentDetectionFilters \(p. 824\)](#)

Obligatorio: No

### [JobTag \(p. 712\)](#)

Identificador que especificas que se devuelve en la notificación de finalización publicada en el tema de Amazon Simple Notification Service. Por ejemplo, puede utilizar `JobTag` para agrupar trabajos relacionados e identificarlos en la notificación de finalización.

Type: Cadena

Restricciones de longitud: Longitud mínima de 1. La longitud máxima es de 256 caracteres.

Patrón: [a-zA-Z0-9\_.\:-]+

Obligatorio: No

### [NotificationChannel \(p. 712\)](#)

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 segmentos. Tenga en cuenta que el tema de Amazon SNS debe tener un nombre de tema que comience por `AmazonRekognition` si utilizas la política de permisos de `AmazonRekognitionServiceRole` para acceder al tema.

Tipo: objeto [NotificationChannel \(p. 799\)](#)

Obligatorio: No

### [SegmentTypes \(p. 712\)](#)

Una serie de tipos de segmentos para detectar en el vídeo. Los valores válidos son `TECHNICAL_CUE` y `SHOT`.

Type: Matriz de cadenas

Miembros de la matriz: Número mínimo de 1 elemento.

Valores válidos: `TECHNICAL_CUE` | `SHOT`

Obligatorio: Sí

### [Video \(p. 712\)](#)

Archivo de vídeo almacenado en un bucket de Amazon S3. Operaciones de inicio de vídeo de Amazon Rekognition, tales como [StartLabelDetection \(p. 701\)](#) usar `Video` para especificar un vídeo para su análisis. Los formatos de archivo compatibles son .mp4, .mov y .avi.

Tipo: objeto [Video \(p. 844\)](#)

Obligatorio: Sí

## Sintaxis de la respuesta

```
{
 "JobId": "string"
}
```

## Elementos de respuesta

Si la acción se realiza correctamente, el servicio devuelve una respuesta HTTP 200.

El servicio devuelve los datos siguientes en formato JSON.

### [JobId \(p. 714\)](#)

Identificador único del trabajo de detección de segmentos. La jobId se devuelve desde `StartSegmentDetection`.

Type: Cadena

Restricciones de longitud: Longitud mínima de 1. La longitud máxima es 64.

Patrón: ^[a-zA-Z0-9-\_]+\$

## Errores

### [AccessDeniedException](#)

No tiene autorización para realizar la acción.

Código de estado HTTP: 400

### [IdempotentParameterMismatchException](#)

[UNAclientRequestToken](#)Se ha reutilizado un parámetro de entrada con una operación, pero al menos uno de los demás parámetros de entrada es distinto de la llamada anterior a la operación.

Código de estado HTTP: 400

### [InternalServerError](#)

Amazon Lex ha tenido un problema de servicio. Pruebe la llamada de nuevo.

Código de estado HTTP: 500

### [InvalidParameterException](#)

El parámetro de entrada infringió una restricción. Valide el parámetro antes de llamar a la operación de la API de nuevo.

Código de estado HTTP: 400

### [InvalidS3ObjectException](#)

Amazon Rekognition no puede obtener acceso al objeto de S3 especificado en la solicitud.

Código de estado HTTP: 400

#### **LimitExceeded**

Se ha superado un límite de servicio de Amazon Rekognition. Por ejemplo, si inicia demasiados trabajos de Amazon Rekognition Video simultáneamente, llama para iniciar operaciones (`StartLabelDetection`, por ejemplo) elevará un `LimitExceeded` excepción (código de estado HTTP: 400) hasta que el número de trabajos ejecutados simultáneamente se encuentre por debajo del límite de servicio de Amazon Rekognition.

Código de estado HTTP: 400

#### **ProvisionedThroughputExceeded**

El número de solicitudes ha superado su límite de rendimiento. Si necesita aumentar este límite, póngase en contacto con Amazon Rekognition.

Código de estado HTTP: 400

#### **ThrottlingException**

Amazon Lex no puede procesar temporalmente la solicitud. Pruebe la llamada de nuevo.

Código de estado HTTP: 500

#### **VideoTooLargeException**

El tamaño del archivo o la duración del medio suministrado es demasiado grande. El tamaño de archivo máximo es de 10 GB. La duración máxima es de 6 horas.

Código de estado HTTP: 400

## Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- [AWS Command Line Interface](#)
- [SDK de AWS para .NET](#)
- [AWS SDK para C++](#)
- [AWS SDK para Go](#)
- [AWSSDK para Java V2](#)
- [AWS SDK para JavaScript](#)
- [SDK de AWS para PHP V3](#)
- [SDK de AWS para Python](#)
- [SDK de AWS para Ruby V3](#)

## StartStreamProcessor

Comienza el procesamiento de un procesador de streaming. Crea un procesador de streaming llamando a [CreateStreamProcessor \(p. 537\)](#). Para contar `StartStreamProcessor` qué procesador de flujo iniciar, utilice el valor de la `Name` campo especificado en la llamada a `CreateStreamProcessor`.

### Sintaxis de la solicitud

```
{
 "Name" : "string"
}
```

### Parámetros de solicitud

La solicitud acepta los siguientes datos en formato JSON.

#### Name (p. 716)

El nombre del procesador de secuencias que se va a iniciar el procesamiento.

Type: Cadena

Restricciones de longitud: Longitud mínima de 1. La longitud máxima es de 128 caracteres.

Patrón: [ a-zA-Z0-9\_.\-\+] +

Obligatorio: Sí

### Elementos de respuesta

Si la acción se realiza correctamente, el servicio devuelve una respuesta HTTP 200 con un cuerpo HTTP vacío.

### Errores

#### AccessDeniedException

No tiene autorización para realizar la acción.

Código de estado HTTP: 400

#### InternalServerError

Amazon Lex ha tenido un problema de servicio. Pruebe la llamada de nuevo.

Código de estado HTTP: 500

#### InvalidParameterException

El parámetro de entrada infringió una restricción. Valide los parámetros antes de llamar a la operación de la API de nuevo.

Código de estado HTTP: 400

#### ProvisionedThroughputExceededException

El número de solicitudes ha superado su límite de rendimiento. Si necesita aumentar este límite, póngase en contacto con Amazon Rekognition.

Código de estado HTTP: 400  
`ResourceInUseException`

El recurso especificado ya se está utilizando.

Código de estado HTTP: 400  
`ResourceNotFoundException`

El recurso especificado en la solicitud no se encuentra.

Código de estado HTTP: 400  
`ThrottlingException`

Amazon Lex no puede procesar temporalmente la solicitud. Pruebe la llamada de nuevo.

Código de estado HTTP: 500

## Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- [AWS Command Line Interface](#)
- [SDK de AWS para .NET](#)
- [AWS SDK para C++](#)
- [AWS SDK para Go](#)
- [AWSSDK para Java V2](#)
- [AWS SDK para JavaScript](#)
- [SDK de AWS para PHP V3](#)
- [SDK de AWS para Python](#)
- [SDK de AWS para Ruby V3](#)

## StartTextDetection

Comienza la detección asincrónica del texto de un vídeo almacenado.

Amazon Rekognition Video puede detectar texto en un vídeo almacenado en un bucket de Amazon S3. Usar[Video \(p. 844\)](#)para especificar el nombre del depósito y el nombre de archivo del vídeo.[StartTextDetection](#)devuelve un identificador de trabajo (`JobId`) que utiliza para obtener los resultados de la operación. Cuando finaliza la detección de texto, Amazon Rekognition Video publica un estado de finalización en el tema Amazon Simple Notification Service que especifique en[NotificationChannel](#).

Para obtener los resultados de la operación de detección de texto, compruebe primero que el valor de estado publicado en el tema de Amazon SNS es `SUCCEEDED`. si es así, llame[GetTextDetection \(p. 640\)](#)y pasa el identificador de trabajo (`JobId`) desde la llamada inicial hasta[StartTextDetection](#).

### Sintaxis de la solicitud

```
{
 "ClientRequestToken": "string",
 "Filters": {
 "RegionsOfInterest": [
 {
 "BoundingBox": {
 "Height": number,
 "Left": number,
 "Top": number,
 "Width": number
 }
 }
],
 "WordFilter": {
 "MinBoundingBoxHeight": number,
 "MinBoundingBoxWidth": number,
 "MinConfidence": number
 }
 },
 "JobTag": "string",
 "NotificationChannel": {
 "RoleArn": "string",
 "SNSTopicArn": "string"
 },
 "Video": {
 "S3Object": {
 "Bucket": "string",
 "Name": "string",
 "Version": "string"
 }
 }
}
```

### Parámetros de solicitud

La solicitud acepta los siguientes datos en formato JSON.

#### [ClientRequestToken \(p. 718\)](#)

Token idempotente utilizado para identificar la solicitud de inicio. Si utilizas el mismo token con varios[StartTextDetection](#)solicitudes, lo mismo `JobId` se devuelve. Usar[ClientRequestToken](#)para evitar que el mismo trabajo se inicie accidentalmente más de una vez.

Type: Cadena

Restricciones de longitud: Longitud mínima de 1. La longitud máxima es 64.

Patrón: ^[a-zA-Z0-9-\_]+\$

Obligatorio: No

#### [Filters \(p. 718\)](#)

Parámetros opcionales que le permiten establecer los criterios que debe cumplir el texto para incluirse en la respuesta.

Tipo: objeto [StartTextDetectionFilters \(p. 827\)](#)

Obligatorio: No

#### [JobTag \(p. 718\)](#)

Obligatorio en el estado de realización publicado por el tema de Amazon Simple Notification Service (SNS). Por ejemplo, puede utilizar JobTag para agrupar trabajos relacionados e identificarlos en la notificación de finalización.

Type: Cadena

Restricciones de longitud: Longitud mínima de 1. La longitud máxima es de 256 caracteres.

Patrón: [a-zA-Z0-9\_.\-\:]+

Obligatorio: No

#### [NotificationChannel \(p. 718\)](#)

El tema de Amazon Rekognition Simple Notification Service (SNS). Para obtener más información, consulte [Llamar a las operaciones de Amazon Rekognition Video \(p. 66\)](#). Tenga en cuenta que el tema de Amazon SNS debe tener un nombre de tema que comience por AmazonRekognitionsi utiliza la política de permisos de AmazonRekognitionServiceRole para acceder al tema. Para obtener más información, consulte [Acceso a varios temas de Amazon SNS](#).

Tipo: objeto [NotificationChannel \(p. 799\)](#)

Obligatorio: No

#### [Video \(p. 718\)](#)

Archivo de vídeo almacenado en un bucket de Amazon S3. Operaciones de inicio de vídeo de Amazon Rekognition, tales como [StartLabelDetection \(p. 701\)](#) usar video para especificar un vídeo para su análisis. Los formatos de archivo compatibles son .mp4, .mov y .avi.

Tipo: objeto [Video \(p. 844\)](#)

Obligatorio: Sí

## Sintaxis de la respuesta

```
{
 "JobId": "string"
}
```

## Elementos de respuesta

Si la acción se realiza correctamente, el servicio devuelve una respuesta HTTP 200.

El servicio devuelve los datos siguientes en formato JSON.

#### [JobId](#) (p. 719)

Identificador del trabajo de detección de texto. Usar `JobId` para identificar el trabajo en una llamada posterior a `GetTextDetection`.

Type: Cadena

Restricciones de longitud: Longitud mínima de 1. La longitud máxima es 64.

Patrón: ^[a-zA-Z0-9-\_]+\$

## Errores

### AccessDeniedException

No tiene autorización para realizar la acción.

Código de estado HTTP: 400

### IdempotentParameterMismatchException

`UNAcientRequestToken`El parámetro de entrada se ha reutilizado con una operación, pero al menos uno de los demás parámetros de entrada es distinto de la llamada anterior a la operación.

Código de estado HTTP: 400

### InternalServerError

Amazon Lex ha tenido un problema de servicio. Pruebe la llamada de nuevo.

Código de estado HTTP: 500

### InvalidParameterException

El parámetro de entrada infringió una restricción. Valide el parámetro antes de llamar a la operación de la API de nuevo.

Código de estado HTTP: 400

### InvalidS3ObjectException

Amazon Rekognition no puede obtener acceso al objeto de S3 especificado en la solicitud.

Código de estado HTTP: 400

### LimitExceededException

Se ha superado un límite de servicio de Amazon Rekognition. Por ejemplo, si inicia demasiados trabajos de Amazon Rekognition Video simultáneamente, llama para iniciar operaciones (`StartLabelDetection`, por ejemplo) elevará un `LimitExceededException` excepción (código de estado HTTP: 400) hasta que el número de trabajos ejecutados simultáneamente se encuentre por debajo del límite de servicio de Amazon Rekognition.

Código de estado HTTP: 400

### ProvisionedThroughputExceededException

El número de solicitudes ha superado su límite de rendimiento. Si necesita aumentar este límite, póngase en contacto con Amazon Rekognition.

Código de estado HTTP: 400

#### ThrottlingException

Amazon Lex no puede procesar temporalmente la solicitud. Pruebe la llamada de nuevo.

Código de estado HTTP: 500

#### VideoTooLargeException

El tamaño del archivo o la duración del medio suministrado es demasiado grande. El tamaño de archivo máximo es de 10 GB. La duración máxima es de 6 horas.

Código de estado HTTP: 400

## Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- [AWS Command Line Interface](#)
- [SDK de AWS para .NET](#)
- [AWS SDK para C++](#)
- [AWS SDK para Go](#)
- [AWSSDK para Java V2](#)
- [AWS SDK para JavaScript](#)
- [SDK de AWS para PHP V3](#)
- [SDK de AWS para Python](#)
- [SDK de AWS para Ruby V3](#)

## StopProjectVersion

Detiene un modelo en ejecución. La operación podría llevar algún tiempo. Para comprobar el estado actual, llame a [DescribeProjectVersions \(p. 564\)](#).

### Sintaxis de la solicitud

```
{
 "ProjectVersionArn": "string"
}
```

### Parámetros de solicitud

La solicitud acepta los siguientes datos en formato JSON.

#### [ProjectVersionArn \(p. 722\)](#)

El nombre de recurso de Amazon (ARN) de la versión del modelo que desea eliminar.

Esta operación requiere permisos para realizar la acción `rekognition:StopProjectVersion`.

Type: Cadena

Restricciones de longitud: Longitud mínima de 20. La longitud máxima es de 2048 caracteres.

Patrón: (^arn:[a-zA-Z\d-]+:rekognition:[a-zA-Z\d-]+\d{12}:project\/[a-zA-Z0-9\_.\-\\_]{1,255}\version\/[a-zA-Z0-9\_.\-\\_]{1,255}\/[0-9]+\$)

Obligatorio: Sí

### Sintaxis de la respuesta

```
{
 "Status": "string"
}
```

### Elementos de respuesta

Si la acción se realiza correctamente, el servicio devuelve una respuesta HTTP 200.

El servicio devuelve los datos siguientes en formato JSON.

#### [Status \(p. 722\)](#)

El estado actual de la operación de parada.

Type: Cadena

Valores válidos: TRAINING\_IN\_PROGRESS | TRAINING\_COMPLETED | TRAINING\_FAILED | STARTING | RUNNING | FAILED | STOPPING | STOPPED | DELETING

## Errores

### [AccessDeniedException](#)

No tiene autorización para realizar la acción.

Código de estado HTTP: 400

InternalServerError

Amazon Lex ha tenido un problema de servicio. Pruebe la llamada de nuevo.

Código de estado HTTP: 500

InvalidParameterException

El parámetro de entrada infringió una restricción. Valide el parámetro antes de llamar a la operación de la API de nuevo.

Código de estado HTTP: 400

ProvisionedThroughputExceededException

El número de solicitudes ha superado su límite de rendimiento. Si necesita aumentar este límite, póngase en contacto con Amazon Rekognition.

Código de estado HTTP: 400

ResourceInUseException

El recurso especificado ya se está utilizando.

Código de estado HTTP: 400

ResourceNotFoundException

El recurso especificado en la solicitud no se encuentra.

Código de estado HTTP: 400

ThrottlingException

Amazon Lex no puede procesar temporalmente la solicitud. Pruebe la llamada de nuevo.

Código de estado HTTP: 500

## Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- [AWS Command Line Interface](#)
- [SDK de AWS para .NET](#)
- [AWS SDK para C++](#)
- [AWS SDK para Go](#)
- [AWSSDK para Java V2](#)
- [AWS SDK para JavaScript](#)
- [SDK de AWS para PHP V3](#)
- [SDK de AWS para Python](#)
- [SDK de AWS para Ruby V3](#)

## StopStreamProcessor

Detiene un procesador de streaming en ejecución creado por[CreateStreamProcessor \(p. 537\)](#).

### Sintaxis de la solicitud

```
{
 "Name": "string"
}
```

### Parámetros de solicitud

La solicitud acepta los siguientes datos en formato JSON.

#### Name (p. 724)

El nombre de un procesador de streaming creado por[CreateStreamProcessor \(p. 537\)](#).

Type: Cadena

Restricciones de longitud: Longitud mínima de 1. La longitud máxima es de 128 caracteres.

Patrón: [a-zA-Z0-9\_.\-\-]+

Obligatorio: Sí

### Elementos de respuesta

Si la acción se realiza correctamente, el servicio devuelve una respuesta HTTP 200 con un cuerpo HTTP vacío.

### Errores

#### AccessDeniedException

No tiene autorización para realizar la acción.

Código de estado HTTP: 400

#### InternalServerError

Amazon Lex ha tenido un problema de servicio. Pruebe la llamada de nuevo.

Código de estado HTTP: 500

#### InvalidArgumentException

El parámetro de entrada infringió una restricción. Valide el parámetro antes de llamar a la operación de la API de nuevo.

Código de estado HTTP: 400

#### ProvisionedThroughputExceededException

El número de solicitudes ha superado su límite de rendimiento. Si necesita aumentar este límite, póngase en contacto con Amazon Rekognition.

Código de estado HTTP: 400

#### ResourceInUseException

El recurso especificado ya se está utilizando.

Código de estado HTTP: 400

#### ResourceNotFoundException

No se encuentra el recurso especificado en la solicitud.

Código de estado HTTP: 400

#### ThrottlingException

Amazon Lex no puede procesar temporalmente la solicitud. Pruebe la llamada de nuevo.

Código de estado HTTP: 500

## Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- [AWS Command Line Interface](#)
- [SDK de AWS para .NET](#)
- [AWS SDK para C++](#)
- [AWS SDK para Go](#)
- [AWSSDK para Java V2](#)
- [AWS SDK para JavaScript](#)
- [SDK de AWS para PHP V3](#)
- [SDK de AWS para Python](#)
- [SDK de AWS para Ruby V3](#)

## TagResource

Añade una o más etiquetas de valor clave a una colección de Amazon Rekognition, un procesador de streaming o un modelo de etiquetas personalizadas. Para obtener más información, consulte [Etiquetado de recursos de AWS](#).

Esta operación requiere permisos para realizar la acción `rekognition:TagResource`.

### Sintaxis de la solicitud

```
{
 "ResourceArn": "string",
 "Tags": {
 "string" : "string"
 }
}
```

### Parámetros de solicitud

La solicitud acepta los siguientes datos en formato JSON.

#### [ResourceArn](#) (p. 726)

Amazon Resource Name (ARN) del modelo, colección o procesador de flujo de al que desea asignar las etiquetas.

Type: Cadena

Restricciones de longitud: Longitud mínima de 20. La longitud máxima es de 2048 caracteres.

Obligatorio: Sí

#### [Tags](#) (p. 726)

Las etiquetas clave-valor para asignar al recurso.

Type: Asignación de cadena a cadena

Entradas de mapa: El número mínimo es 0 elementos. Número máximo de 200 elementos.

Restricciones de longitud clave: Longitud mínima de 1. La longitud máxima es de 128 caracteres.

Patrón de clave: ^(?!aws:)[\p{L}\p{Z}\p{N}\_.:/=+\-\@]\*\$

Restricciones de longitud de valor: Longitud mínima de 0. La longitud máxima es de 256 caracteres.

Pattern de valores: ^([\p{L}\p{Z}\p{N}\_.:/=+\-\@]\*)\$

Obligatorio: Sí

### Elementos de respuesta

Si la acción se realiza correctamente, el servicio devuelve una respuesta HTTP 200 con un cuerpo HTTP vacío.

### Errores

#### [AccessDeniedException](#)

No tiene autorización para realizar la acción.

Código de estado HTTP: 400  
InternalServerError

Amazon Lex ha tenido un problema de servicio. Pruebe la llamada de nuevo.

Código de estado HTTP: 500  
InvalidParameterException

El parámetro de entrada infringió una restricción. Valide el parámetro antes de llamar a la operación de la API de nuevo.

Código de estado HTTP: 400  
ProvisionedThroughputExceededException

El número de solicitudes ha superado su límite de rendimiento. Si necesita aumentar este límite, póngase en contacto con Amazon Rekognition.

Código de estado HTTP: 400  
ResourceNotFoundException

No se encuentra el recurso especificado en la solicitud.

Código de estado HTTP: 400  
ServiceQuotaExceededException

El tamaño del recurso supera el límite permitido. Para obtener más información, consulte [Directrices y cuotas de Amazon Rekognition \(p. 847\)](#).

Código de estado HTTP: 400  
ThrottlingException

Amazon Lex no puede procesar temporalmente la solicitud. Pruebe la llamada de nuevo.

Código de estado HTTP: 500

## Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- [AWS Command Line Interface](#)
- [SDK de AWS para .NET](#)
- [AWS SDK para C++](#)
- [AWS SDK para Go](#)
- [AWSSDK para Java V2](#)
- [AWS SDK para JavaScript](#)
- [SDK de AWS para PHP V3](#)
- [SDK de AWS para Python](#)
- [SDK de AWS para Ruby V3](#)

## UntagResource

Elimina una o más etiquetas de una colección de Amazon Rekognition, procesador de flujo de o modelo de etiquetas personalizadas.

Esta operación requiere permisos para realizar la acción `rekognition:UntagResource`.

### Sintaxis de la solicitud

```
{
 "ResourceArn": "string",
 "TagKeys": ["string"]
}
```

### Parámetros de solicitud

La solicitud acepta los siguientes datos en formato JSON.

#### [ResourceArn \(p. 728\)](#)

Nombre de recurso de Amazon (ARN) del modelo, colección de o procesador de flujo del que desea eliminar las etiquetas.

Type: Cadena

Restricciones de longitud: Longitud mínima de 20. La longitud máxima es de 2048 caracteres.

obligatorio: obligatorio Sí

#### [TagKeys \(p. 728\)](#)

Una lista de las etiquetas que desea eliminar.

Type: Matriz de cadenas

Miembros de matriz: El número mínimo es 0 elementos. Número máximo de 200 elementos.

Restricciones de longitud: Longitud mínima de 1. La longitud máxima es de 128 caracteres.

Patrón: ^(?!aws:)[\p{L}\p{Z}\p{N}\_.:/+=\-\@]\*\$

obligatorio: obligatorio Sí

### Elementos de respuesta

Si la acción se realiza correctamente, el servicio devuelve una respuesta HTTP 200 con un cuerpo HTTP vacío.

### Errores

#### `AccessDeniedException`

No tiene autorización para realizar la acción.

Código de estado HTTP 400

#### `InternalServerError`

Amazon Lex ha tenido un problema de servicio. Pruebe la llamada de nuevo.

Código de estado HTTP 500

InvalidParameterException

El parámetro de entrada infringió una restricción. Valide el parámetro antes de llamar a la operación de la API de nuevo.

Código de estado HTTP 400

ProvisionedThroughputExceededException

El número de solicitudes ha superado su límite de rendimiento. Si necesita aumentar este límite, póngase en contacto con Amazon Rekognition.

Código de estado HTTP 400

ResourceNotFoundException

No se encuentra el recurso especificado en la solicitud.

Código de estado HTTP 400

ThrottlingException

Amazon Lex no puede procesar temporalmente la solicitud. Pruebe la llamada de nuevo.

Código de estado HTTP 500

## Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- [AWS Command Line Interface](#)
- [SDK de AWS para .NET](#)
- [AWS SDK para C++](#)
- [AWS SDK para Go](#)
- [AWSSDK para Java V2](#)
- [AWS SDK para JavaScript](#)
- [SDK de AWS para PHP V3](#)
- [SDK de AWS para Python](#)
- [SDK de AWS para Ruby V3](#)

## UpdateDatasetEntries

Añade o actualiza una o más entradas (imágenes) de un conjunto de datos. Una entrada es una línea JSON que contiene la información de una sola imagen, incluida la ubicación de la imagen, las etiquetas asignadas y los cuadros delimitadores de ubicación de objetos. Para obtener más información, consulte [Etiquetas a nivel de imagen en archivos de manifiesto](#) y [Localización de objetos en archivos de manifiesto](#).

Si el archivo `desource-ref` de la línea JSON hace referencia a una imagen existente, la imagen existente del conjunto de datos se actualiza. Si `source-ref` hace referencia a una imagen existente, la imagen se añade como una nueva imagen al conjunto de datos.

Se especifican los cambios que desea realizar en el `changes` Parámetro de entrada. No hay límite para el número de líneas JSON que puedes cambiar, sino el tamaño de `changes` Debe ser menor que 5 MB.

`UpdateDatasetEntries` Devuelve inmediatamente, pero la actualización del conjunto de datos puede tardar un tiempo en completarse. Usar [DescribeDataset \(p. 559\)](#) para comprobar el estado actual. El conjunto de datos se ha actualizado correctamente si el valor de `status` es `UPDATE_COMPLETE`.

Para comprobar si se ha producido algún error no terminal, llame [ListDatasetEntries \(p. 656\)](#) y compruebe la presencia de errores en las líneas JSON.

La actualización del conjunto de datos falla si se produce un error de terminal (`Status=UPDATE_FAILED`). En este momento, no se puede acceder a la información de error de terminal desde el SDK de etiquetas personalizadas de Amazon Rekognition.

Esta operación requiere permisos para realizar la acción `rekognition:UpdateDatasetEntries`.

## Sintaxis de la solicitud

```
{
 "Changes": {
 "GroundTruth": "blob"
 },
 "DatasetArn": "string"
}
```

## Parámetros de solicitud

La solicitud acepta los siguientes datos en formato JSON.

### [Changes \(p. 730\)](#)

Los cambios que desea realizar en el conjunto de datos.

Tipo: objeto [DatasetChanges \(p. 754\)](#)

Obligatorio: Sí

### [DatasetArn \(p. 730\)](#)

El nombre de recurso de Amazon (ARN) del conjunto de datos que desea actualizar.

Type: Cadena

Restricciones de longitud: Longitud mínima de 20. La longitud máxima es de 2048 caracteres.

Patrón: (^arn:[a-z\d-]+:rekognition:[a-z\d-]+:\d{12}:project\/[a-zA-Z0-9\_.\-\\_]{1,255}\dataset\/(train|test)\/[0-9]+\\$)

Obligatorio: Sí

## Elementos de respuesta

Si la acción se realiza correctamente, el servicio devuelve una respuesta HTTP 200 con un cuerpo HTTP vacío.

## Errores

AccessDeniedException

No tiene autorización para realizar la acción.

Código de estado HTTP: 400

InternalServerError

Amazon Lex ha tenido un problema de servicio. Pruebe la llamada de nuevo.

Código de estado HTTP: 500

InvalidArgumentException

El parámetro de entrada infringió una restricción. Valide el parámetro antes de llamar a la operación de la API de nuevo.

Código de estado HTTP: 400

LimitExceededException

Se ha superado un límite de servicio de Amazon Rekognition. Por ejemplo, si inicia demasiados trabajos de Amazon Rekognition Video simultáneamente, llama para iniciar operaciones (`StartLabelDetection`, por ejemplo) elevará un `LimitExceededException` excepción (código de estado HTTP: 400) hasta que el número de trabajos ejecutados simultáneamente se encuentre por debajo del límite de servicio de Amazon Rekognition.

Código de estado HTTP: 400

ProvisionedThroughputExceededException

El número de solicitudes ha superado su límite de rendimiento. Si necesita aumentar este límite, póngase en contacto con Amazon Rekognition.

Código de estado HTTP: 400

ResourceInUseException

El recurso especificado ya se está utilizando.

Código de estado HTTP: 400

ResourceNotFoundException

El recurso especificado en la solicitud no se encuentra.

Código de estado HTTP: 400

ThrottlingException

Amazon Lex no puede procesar temporalmente la solicitud. Pruebe la llamada de nuevo.

Código de estado HTTP: 500

## Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- [AWS Command Line Interface](#)
- [SDK de AWS para .NET](#)
- [AWS SDK para C++](#)
- [AWS SDK para Go](#)
- [AWSSDK para Java V2](#)
- [AWS SDK para JavaScript](#)
- [SDK de AWS para PHP V3](#)
- [SDK de AWS para Python](#)
- [SDK de AWS para Ruby V3](#)

## Tipos de datos

Los tipos de datos siguientes son compatibles:

- [AgeRange \(p. 735\)](#)
- [Asset \(p. 736\)](#)
- [AudioMetadata \(p. 737\)](#)
- [Beard \(p. 738\)](#)
- [BlackFrame \(p. 739\)](#)
- [BoundingBox \(p. 740\)](#)
- [Celebrity \(p. 742\)](#)
- [CelebrityDetail \(p. 744\)](#)
- [CelebrityRecognition \(p. 746\)](#)
- [ComparedFace \(p. 747\)](#)
- [ComparedSourceImageFace \(p. 749\)](#)
- [CompareFacesMatch \(p. 750\)](#)
- [ContentModerationDetection \(p. 751\)](#)
- [CoversBodyPart \(p. 752\)](#)
- [CustomLabel \(p. 753\)](#)
- [DatasetChanges \(p. 754\)](#)
- [DatasetDescription \(p. 755\)](#)
- [DatasetLabelDescription \(p. 757\)](#)
- [DatasetLabelStats \(p. 758\)](#)
- [DatasetMetadata \(p. 759\)](#)
- [DatasetSource \(p. 761\)](#)
- [DatasetStats \(p. 762\)](#)
- [DetectionFilter \(p. 763\)](#)
- [DetectTextFilters \(p. 764\)](#)
- [DistributeDataset \(p. 765\)](#)
- [Emotion \(p. 766\)](#)
- [EquipmentDetection \(p. 767\)](#)

- [EvaluationResult \(p. 768\)](#)
- [Eyeglasses \(p. 769\)](#)
- [EyeOpen \(p. 770\)](#)
- [Face \(p. 771\)](#)
- [FaceDetail \(p. 773\)](#)
- [FaceDetection \(p. 776\)](#)
- [FaceMatch \(p. 777\)](#)
- [FaceRecord \(p. 778\)](#)
- [FaceSearchSettings \(p. 779\)](#)
- [Gender \(p. 780\)](#)
- [Geometry \(p. 781\)](#)
- [GroundTruthManifest \(p. 782\)](#)
- [HumanLoopActivationOutput \(p. 783\)](#)
- [HumanLoopConfig \(p. 784\)](#)
- [HumanLoopDataAttributes \(p. 785\)](#)
- [Image \(p. 786\)](#)
- [ImageQuality \(p. 788\)](#)
- [Instance \(p. 789\)](#)
- [KinesisDataStream \(p. 790\)](#)
- [KinesisVideoStream \(p. 791\)](#)
- [KnownGender \(p. 792\)](#)
- [Label \(p. 793\)](#)
- [LabelDetection \(p. 794\)](#)
- [Landmark \(p. 795\)](#)
- [ModerationLabel \(p. 796\)](#)
- [MouthOpen \(p. 797\)](#)
- [Mustache \(p. 798\)](#)
- [NotificationChannel \(p. 799\)](#)
- [OutputConfig \(p. 800\)](#)
- [Parent \(p. 801\)](#)
- [PersonDetail \(p. 802\)](#)
- [PersonDetection \(p. 803\)](#)
- [PersonMatch \(p. 804\)](#)
- [Point \(p. 805\)](#)
- [Pose \(p. 806\)](#)
- [ProjectDescription \(p. 807\)](#)
- [ProjectVersionDescription \(p. 808\)](#)
- [ProtectiveEquipmentBodyPart \(p. 811\)](#)
- [ProtectiveEquipmentPerson \(p. 812\)](#)
- [ProtectiveEquipmentSummarizationAttributes \(p. 813\)](#)
- [ProtectiveEquipmentSummary \(p. 814\)](#)
- [RegionOfInterest \(p. 816\)](#)
- [S3Object \(p. 817\)](#)
- [SegmentDetection \(p. 818\)](#)
- [SegmentTypeInfo \(p. 821\)](#)
- [ShotSegment \(p. 822\)](#)

- [Smile \(p. 823\)](#)
- [StartSegmentDetectionFilters \(p. 824\)](#)
- [StartShotDetectionFilter \(p. 825\)](#)
- [StartTechnicalCueDetectionFilter \(p. 826\)](#)
- [StartTextDetectionFilters \(p. 827\)](#)
- [StreamProcessor \(p. 828\)](#)
- [StreamProcessorInput \(p. 829\)](#)
- [StreamProcessorOutput \(p. 830\)](#)
- [StreamProcessorSettings \(p. 831\)](#)
- [Summary \(p. 832\)](#)
- [Sunglasses \(p. 833\)](#)
- [TechnicalCueSegment \(p. 834\)](#)
- [TestingData \(p. 835\)](#)
- [TestingDataResult \(p. 836\)](#)
- [TextDetection \(p. 837\)](#)
- [TextDetectionResult \(p. 839\)](#)
- [TrainingData \(p. 840\)](#)
- [TrainingDataResult \(p. 841\)](#)
- [UnindexedFace \(p. 842\)](#)
- [ValidationData \(p. 843\)](#)
- [Video \(p. 844\)](#)
- [VideoMetadata \(p. 845\)](#)

## AgeRange

Estructura que contiene una estimación del rango de edades, en años, para una cara.

Amazon Rekognition estima una estimación del rango de edades para los rostros detectados en la imagen de entrada. Los rangos de edad estimados pueden superponerse. Una cara de un niño de 5 años podría tener un rango estimado de 4-6, mientras que la cara de un niño de 6 años podría tener un rango estimado de 4 a 8.

### Contenido

High

La edad estimada más alta.

Type: Entero

Rango válido: Valor mínimo de 0.

Obligatorio: No

Low

La edad estimada más baja.

Type: Entero

Rango válido: Valor mínimo de 0.

Obligatorio: No

### Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- [AWS SDK para C++](#)
- [AWS SDK para Go](#)
- [AWSSDK para Java V2](#)
- [SDK de AWS para Ruby V3](#)

## Asset

Los activos son las imágenes que utiliza para entrenar y evaluar una versión de modelo. Los activos también pueden contener información de validación que se utiliza para depurar una formación de modelo fallida.

### Contenido

GroundTruthManifest

El bucket de S3 que contiene un archivo de manifiesto de Amazon Sagemaker Ground Truth.

Tipo: objeto [GroundTruthManifest \(p. 782\)](#)

Obligatorio: No

### Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- [AWS SDK para C++](#)
- [AWS SDK para Go](#)
- [AWSSDK para Java V2](#)
- [SDK de AWS para Ruby V3](#)

## AudioMetadata

Información de metadatos sobre una transmisión de audio. Una matriz de AudioMetadata los objetos de las secuencias de audio que se encuentran en un vídeo almacenado son devueltos por [GetSegmentDetection \(p. 635\)](#).

### Contenido

#### Codec

Códec de audio utilizado para codificar o decodificar la transmisión de audio.

Type: Cadena

Obligatorio: No

#### DurationMillis

La duración de la transmisión de audio en milisegundos.

Type: Long

Rango válido: Valor mínimo de 0.

Obligatorio: No

#### NumberOfChannels

El número de canales de audio del segmento.

Type: Long

Rango válido: Valor mínimo de 0.

Obligatorio: No

#### SampleRate

Frecuencia de muestreo de la transmisión de audio.

Type: Long

Rango válido: Valor mínimo de 0.

Obligatorio: No

## Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- [AWS SDK para C++](#)
- [AWS SDK para Go](#)
- [AWSSDK para Java V2](#)
- [SDK de AWS para Ruby V3](#)

## Beard

Indica si la cara tiene barba o no y el nivel de confianza en la determinación.

### Contenido

#### Confidence

Nivel de confianza en la determinación.

Type: Float

Rango válido: Valor mínimo de 0. Valor máximo de 100.

Obligatorio: No

#### Value

: un valor booleano que indica si la cara tiene barba o no.

Type: Booleano

Obligatorio: No

### Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- [AWS SDK para C++](#)
- [AWS SDK para Go](#)
- [AWSSDK para Java V2](#)
- [SDK de AWS para Ruby V3](#)

## BlackFrame

Filtro que permite controlar la detección de fotogramas negros especificando los niveles de negro y la cobertura de píxeles de los píxeles negros de un fotograma. Dado que los vídeos pueden provenir de varias fuentes, formatos y períodos de tiempo, pueden contener estándares diferentes y niveles de ruido variables para los fotogramas negros que deben tenerse en cuenta. Para obtener más información, consulte [StartSegmentDetection \(p. 712\)](#).

### Contenido

#### MaxPixelThreshold

Umbral utilizado para determinar el valor máximo de luminancia de un píxel que se considera negro. En un vídeo de gama completa de colores, los valores de luminancia oscilan entre 0 y 255. Un valor de píxel de 0 es negro puro y el filtro más estricto. El valor máximo de píxel negro se calcula de la siguiente manera:  $\text{max\_black\_pixel\_value} = \text{minimum\_luminance} + \text{maxPixelThreshold} * \text{luminance\_range}$ .

Por ejemplo, para un vídeo de rango completo con BlackPixelThreshold = 0,1, max\_black\_pixel\_value es  $0 + 0,1 * (255-0) = 25,5$ .

El valor predeterminado de MaxPixelThreshold es 0,2, que se asigna a un max\_black\_pixel\_value de 51 para un vídeo de rango completo. Puedes bajar este umbral para ser más estricto en los niveles negros.

Type: Float

Rango válido: Valor mínimo de 0. Valor máximo de 1.

Obligatorio: No

#### MinCoveragePercentage

El porcentaje mínimo de píxeles de un fotograma que necesita tener una luminancia por debajo del max\_black\_pixel\_value para que un fotograma se considere un marco negro. La luminancia se calcula utilizando la matriz BT.709.

El valor predeterminado es 99, lo que significa que al menos el 99% de todos los píxeles del marco son píxeles negros según el MaxPixelThresholdconjunto. Puede reducir este valor para permitir más ruido en el marco negro.

Type: Float

Rango válido: Valor mínimo de 0. Valor máximo de 100.

Obligatorio: No

### Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- [AWS SDK para C++](#)
- [AWS SDK para Go](#)
- [AWSSDK para Java V2](#)
- [SDK de AWS para Ruby V3](#)

## BoundingBox

Identifica el cuadro delimitador alrededor de la etiqueta, la cara, el texto o el equipo de protección personal. `Left`(coordenada x) y `Top`(coordenada y) son coordenadas que representan los lados superior e izquierdo del cuadro delimitador. Observe que la esquina superior izquierda de la imagen es el origen (0,0).

`Left`Los valores devueltos son ratios del tamaño de imagen global. 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 `Left`valor de 0,5 (350/700) y `Top`valor de 0,25 (50/200).

`Width`Los valores representan las dimensiones del cuadro delimitador expresada proporcionalmente respecto a la dimensión total de la imagen. Por ejemplo, si la imagen de entrada es de 700 x 200 píxeles y el ancho del cuadro delimitador es de 70 píxeles, el ancho devuelto es 0,1.

### Note

Las coordenadas del cuadro delimitador pueden tener valores negativos. Por ejemplo, si Amazon Rekognition puede detectar una cara que se encuentra en el borde de la imagen y solo está parcialmente visible, el servicio puede devolver coordenadas que se encuentran fuera de los límites de la imagen y, según el borde de la imagen, puede obtener valores negativos o valores superiores a 1 para el `Left`o `Top`Valores.

## Contenido

### Height

Height del cuadro delimitador expresada proporcionalmente respecto a la altura total de la imagen.

Type: Float

Obligatorio: No

### Left

Coordenada izquierda del cuadro delimitador expresada proporcionalmente respecto a la anchura total de la imagen.

Type: Float

Obligatorio: No

### Top

La coordenada superior del cuadro delimitador expresada proporcionalmente respecto a la altura total de la imagen.

Type: Float

Obligatorio: No

### Width

La anchura del cuadro delimitador expresada proporcionalmente respecto a la anchura total de la imagen.

Type: Float

Obligatorio: No

## Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- [AWS SDK para C++](#)
- [AWS SDK para Go](#)
- [AWSSDK para Java V2](#)
- [SDK de AWS para Ruby V3](#)

## Celebrity

Proporciona información sobre una celebridad reconocida por el [RecognizeCelebrities](#) (p. 671).

### Contenido

#### Face

Proporciona información sobre el rostro de la celebridad, como su ubicación en la imagen.

Tipo: objeto [ComparedFace](#) (p. 747)

Obligatorio: No

#### Id

Un identificador único para la celebridad.

Type: Cadena

Patrón: [ 0-9A-Za-z ]\*

Obligatorio: No

#### KnownGender

La identidad de género conocida de la celebridad que coincide con la identificación proporcionada. La identidad de género conocida puede ser hombre, mujer, no binaria o no incluida en la lista.

Tipo: objeto [KnownGender](#) (p. 792)

Obligatorio: No

#### MatchConfidence

La confianza, en porcentaje, de que Amazon Rekognition tiene de que la cara reconocida es la celebridad.

Type: Float

Rango válido: Valor mínimo de 0. Valor máximo de 100.

Obligatorio: No

#### Name

El nombre de la celebridad.

Type: Cadena

Obligatorio: No

#### Urls

Una serie de URL que apuntan a información adicional sobre la celebridad. Si no hay información adicional sobre la celebridad, esta lista está vacía.

Type: Matriz de cadenas

Miembros de matrices: El número mínimo es 0 elementos. Número máximo de 255 elementos.

Obligatorio: No

## Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- [AWS SDK para C++](#)
- [AWS SDK para Go](#)
- [AWSSDK para Java V2](#)
- [SDK de AWS para Ruby V3](#)

## CelebrityDetail

Información sobre un famoso reconocido.

### Contenido

#### BoundingBox

Caja delimitadora alrededor del cuerpo de una celebridad.

Tipo: objeto [BoundingBox](#) (p. 740)

: obligatorio No

#### Confidence

La confianza, en porcentaje, de que Amazon Rekognition tiene de que la cara reconocida es la celebridad.

Type: Float

Rango válido: Valor mínimo de 0. Valor máximo de 100.

: obligatorio No

#### Face

Detalles de la cara de la famosa celebridad.

Tipo: objeto [FaceDetail](#) (p. 773)

: obligatorio No

#### Id

El identificador único para el famoso.

Type: Cadena

Patrón: [ 0-9A-Za-z ]\*

: obligatorio No

#### KnownGender

Recupera el género conocido de la celebridad.

Tipo: objeto [KnownGender](#) (p. 792)

: obligatorio No

#### Name

El nombre de la celebridad.

Type: Cadena

: obligatorio No

#### Urls

Una serie de URL que apuntan a información adicional de celebridades.

Type: Matriz de cadenas

Miembros de matriz: El número mínimo es 0 elementos. Número máximo de 255 elementos.

: obligatorio No

## Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- [AWS SDK para C++](#)
- [AWS SDK para Go](#)
- [AWSSDK para Java V2](#)
- [SDK de AWS para Ruby V3](#)

## CelebrityRecognition

Información sobre una celebridad detectada y el momento en que se detectó la celebridad en un vídeo almacenado. Para obtener más información, consulte [GetCelebrityRecognition \(p. 605\)](#).

### Contenido

#### Celebrity

Información sobre un famoso reconocido.

Tipo: objeto [CelebrityDetail \(p. 744\)](#)

Obligatorio: No

#### Timestamp

El tiempo, en milisegundos desde el comienzo del vídeo, cuando se reconoció a la famosa.

Type: Long

Obligatorio: No

### Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- [AWS SDK para C++](#)
- [AWS SDK para Go](#)
- [AWSSDK para Java V2](#)
- [SDK de AWS para Ruby V3](#)

## ComparedFace

Proporciona metadatos de cara para caras de imagen de destino analizadas por `CompareFaces` y `RecognizeCelebrities`.

### Contenido

#### BoundingBox

Caja delimitadora de la cara.

Tipo: objeto [BoundingBox](#) (p. 740)

Obligatorio: No

#### Confidence

Grado de confianza de que el cuadro delimitador contiene un rostro.

Type: Float

Rango válido: Valor mínimo de 0. Valor máximo de 100.

Obligatorio: No

#### Emotions

Las emociones que parecen expresarse en la cara y el nivel de confianza en la determinación. Los valores válidos incluyen «Feliz», «Triste», «Enfadado», «Confundido», «Disgustado», «Sorprendido», «Calma», «Desconocido» y «Miedo».

Type: Matriz de [Emotion](#) (p. 766) objects

Obligatorio: No

#### Landmarks

Una matriz de referencias faciales.

Type: Matriz de [Landmark](#) (p. 795) objects

Obligatorio: No

#### Pose

Indica la postura del rostro tal como determina su cabeceo, balanceo y desviación.

Tipo: objeto [Pose](#) (p. 806)

Obligatorio: No

#### Quality

Identifica el brillo y la nitidez de la imagen del rostro.

Tipo: objeto [ImageQuality](#) (p. 788)

Obligatorio: No

#### Smile

Indica si la cara sonríe o no y el nivel de confianza en la determinación.

Tipo: objeto [Smile](#) (p. 823)

Obligatorio: No

## Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- [AWS SDK para C++](#)
- [AWS SDK para Go](#)
- [AWSSDK para Java V2](#)
- [SDK de AWS para Ruby V3](#)

## ComparedSourceImageFace

Tipo que describe la cara que Amazon Rekognition eligió para comparar con las caras del objetivo. Contiene un cuadro delimitador para la cara seleccionada y grado de confianza de que el cuadro delimitador contiene un rostro. Tenga en cuenta que Amazon Rekognition selecciona la cara más grande de la imagen de origen para esta comparación.

### Contenido

#### BoundingBox

Caja delimitadora de la cara.

Tipo: objeto [BoundingBox \(p. 740\)](#)

Obligatorio: No

#### Confidence

Grado de confianza de que el cuadro delimitador seleccionado contiene un rostro.

Type: Float

Rango válido: Valor mínimo de 0. Valor máximo de 100.

Obligatorio: No

### Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- [AWS SDK para C++](#)
- [AWS SDK para Go](#)
- [AWSSDK para Java V2](#)
- [SDK de AWS para Ruby V3](#)

## CompareFacesMatch

Proporciona información sobre una cara de una imagen de destino que coincide con la cara de la imagen de origen analizada por `CompareFaces`. La `Face` contiene el cuadro delimitador de la cara de la imagen de destino. La `Similarity` es la confianza de que la cara de la imagen de origen coincide con la cara del cuadro delimitador.

### Contenido

#### Face

Proporciona metadatos de rostro (el delimitador y la confianza de que el delimitador contenga realmente un rostro).

Tipo: objeto [ComparedFace \(p. 747\)](#)

Obligatorio: No

#### Similarity

Nivel de confianza de que los rostros coinciden.

Type: Float

Rango válido: Valor mínimo de 0. Valor máximo de 100.

Obligatorio: No

### Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- [AWS SDK para C++](#)
- [AWS SDK para Go](#)
- [AWSSDK para Java V2](#)
- [SDK de AWS para Ruby V3](#)

## ContentModerationDetection

Información sobre una detección de etiquetas de contenido inapropiada, no deseada u ofensiva en un vídeo almacenado.

### Contenido

#### ModerationLabel

Etiqueta de moderación de contenido detectada por en el vídeo almacenado.

Tipo: objeto [ModerationLabel \(p. 796\)](#)

Obligatorio: No

#### Timestamp

Tiempo, en milisegundos desde el principio del vídeo, cuando se detectó la etiqueta de moderación de contenido.

Type: Long

Obligatorio: No

### Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- [AWS SDK para C++](#)
- [AWS SDK para Go](#)
- [AWSSDK para Java V2](#)
- [SDK de AWS para Ruby V3](#)

## CoversBodyPart

Información sobre un artículo del equipo de protección personal que cubre una parte del cuerpo correspondiente. Para obtener más información, consulte [DetectProtectiveEquipment \(p. 592\)](#).

### Contenido

#### Confidence

La confianza que Amazon Rekognition tiene en el valor de `Value`.

Type: Float

Rango válido: Valor mínimo de 0. Valor máximo de 100.

Obligatorio: No

#### Value

True si el EPP cubre la parte del cuerpo correspondiente, de lo contrario es falso.

Type: Booleano

Obligatorio: No

### Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- [AWS SDK para C++](#)
- [AWS SDK para Go](#)
- [AWSSDK para Java V2](#)
- [SDK de AWS para Ruby V3](#)

## CustomLabel

Etiqueta personalizada detectada en una imagen mediante una llamada a [DetectCustomLabels \(p. 573\)](#).

### Contenido

#### Confidence

La confianza que tiene el modelo en la detección de la etiqueta personalizada. El rango va de 0 a 100. Un valor más elevado implica una mayor confianza.

Type: Float

Rango válido: Valor mínimo de 0. Valor máximo de 100.

Obligatorio: No

#### Geometry

Ubicación del objeto detectado en la imagen que corresponde a la etiqueta personalizada. Incluye un cuadro delimitador grueso alineado con ejes que rodea el objeto y un polígono de grano más fino para obtener información espacial más precisa.

Tipo: objeto [Geometry \(p. 781\)](#)

Obligatorio: No

#### Name

Nombre de la etiqueta personalizada.

Type: Cadena

Obligatorio: No

### Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- [AWS SDK para C++](#)
- [AWS SDK para Go](#)
- [AWSSDK para Java V2](#)
- [SDK de AWS para Ruby V3](#)

## DatasetChanges

Describe las actualizaciones o adiciones a un conjunto de datos. Una actualización o adición única es una entrada (línea JSON) que proporciona información sobre una sola imagen. Para actualizar una entrada existente, coincide con la `source-ref` de la entrada de actualización con el `source-ref` archivado de la entrada que desea actualizar. Si el archivo `desource-ref` coincide con una entrada existente, la entrada se agrega al conjunto de datos como una nueva entrada.

### Contenido

#### GroundTruth

Objeto de datos binario codificado en Base64 que contiene una o líneas JSON que actualizan el conjunto de datos o son adiciones al conjunto de datos. Cambia un conjunto de datos llamando [UpdateDatasetEntries \(p. 730\)](#). Si utiliza un SDK de AWS para llamar `UpdateDatasetEntries`, no es necesario codificar `Changes` ya que el SDK codifica los datos por ti.

Por ejemplo, líneas JSON, consulte [Etiquetas a nivel de imagen en archivos de manifiesto](#) y [Localización de objetos en archivos de manifiesto](#).

Type: Objeto de datos binarios codificados en Base64

Restricciones de longitud: Longitud mínima de 1. La longitud máxima es de 5242880 caracteres.

Obligatorio: Sí

### Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- [AWS SDK para C++](#)
- [AWS SDK para Go](#)
- [AWSSDK para Java V2](#)
- [SDK de AWS para Ruby V3](#)

## DatasetDescription

Descripción de un conjunto de datos. Para obtener más información, consulte [DescribeDataset \(p. 559\)](#).

Los campos de estado `Status`, `StatusMessage`, y `StatusMessageCode` refleja la última operación del conjunto de datos.

### Contenido

#### CreationTimestamp

Marca de hora de Unix para la hora y la fecha en que se creó el conjunto de datos.

Type: Marca temporal

Obligatorio: No

#### DatasetStats

El código de mensaje de estado del conjunto de datos.

Tipo: objeto [DatasetStats \(p. 762\)](#)

Obligatorio: No

#### LastUpdatedTimestamp

La marca temporal de Unix de la fecha y hora en la que se actualizó por última vez el conjunto de datos.

Type: Marca temporal

Obligatorio: No

#### Status

Estado del conjunto de datos.

Type: Cadena

Valores válidos: `CREATE_IN_PROGRESS` | `CREATE_COMPLETE` | `CREATE_FAILED` |  
`UPDATE_IN_PROGRESS` | `UPDATE_COMPLETE` | `UPDATE_FAILED` | `DELETE_IN_PROGRESS`

Obligatorio: No

#### StatusMessage

El mensaje de estado del conjunto de datos.

Type: Cadena

Obligatorio: No

#### StatusMessageCode

El código de mensaje de estado de la operación del conjunto de datos. Si se produce un error de servicio, intente volver a llamar a la API más tarde. Si se produce un error de cliente, compruebe los parámetros de entrada de la llamada a la API del conjunto de datos que ha fallado.

Type: Cadena

Valores válidos: `SUCCESS` | `SERVICE_ERROR` | `CLIENT_ERROR`

Obligatorio: No

## Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- [AWS SDK para C++](#)
- [AWS SDK para Go](#)
- [AWSSDK para Java V2](#)
- [SDK de AWS para Ruby V3](#)

## DatasetLabelDescription

Describe una etiqueta de conjunto de datos. Para obtener más información, consulte [ListDatasetLabels \(p. 660\)](#).

### Contenido

#### LabelName

Nombre de la etiqueta.

Type: Cadena

Restricciones de longitud: Longitud mínima de 1. La longitud máxima es de 255 caracteres.

Patrón: .{1,}

Obligatorio: No

#### LabelStats

Estadísticas sobre la etiqueta.

Tipo: objeto [DatasetLabelStats \(p. 758\)](#)

Obligatorio: No

### Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- [AWS SDK para C++](#)
- [AWS SDK para Go](#)
- [AWSSDK para Java V2](#)
- [SDK de AWS para Ruby V3](#)

## DatasetLabelStats

Estadísticas sobre una etiqueta utilizada en un conjunto de datos. Para obtener más información, consulte [DatasetLabelDescription \(p. 757\)](#).

### Contenido

#### BoundingBoxCount

Número total de imágenes que tienen la etiqueta asignada a un cuadro delimitador.

Type: Entero

Rango válido: Valor mínimo de 0.

Obligatorio: No

#### EntryCount

El número total de imágenes que utilizan la etiqueta.

Type: Entero

Rango válido: Valor mínimo de 0.

Obligatorio: No

### Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- [AWS SDK para C++](#)
- [AWS SDK para Go](#)
- [AWSSDK para Java V2](#)
- [SDK de AWS para Ruby V3](#)

## DatasetMetadata

Información resumida de un conjunto de datos de etiquetas personalizadas de Amazon Rekognition. Para obtener más información, consulte [ProjectDescription \(p. 807\)](#).

### Contenido

#### CreationTimestamp

La marca temporal de Unix de la fecha y hora en que se creó el conjunto de datos.

Type: Marca temporal

Obligatorio: No

#### DatasetArn

El nombre de recurso de Amazon (ARN) del conjunto de datos.

Type: Cadena

Restricciones de longitud: Longitud mínima de 20. La longitud máxima es de 2048 caracteres.

Patrón: (^arn:[a-z\d-]+:rekognition:[a-z\d-]+\d{12}:project\/[a-zA-Z0-9\_.\-\\_]{1,255}\dataset\/(train|test)\/[0-9]+\$)

Obligatorio: No

#### DatasetType

Tipo de conjunto de datos.

Type: Cadena

Valores válidos: TRAIN | TEST

Obligatorio: No

#### Status

El estado del conjunto de datos.

Type: Cadena

Valores válidos: CREATE\_IN\_PROGRESS | CREATE\_COMPLETE | CREATE\_FAILED | UPDATE\_IN\_PROGRESS | UPDATE\_COMPLETE | UPDATE\_FAILED | DELETE\_IN\_PROGRESS

Obligatorio: No

#### StatusMessage

El mensaje de estado del conjunto de datos.

Type: Cadena

Obligatorio: No

#### StatusMessageCode

El código de mensaje de estado de la operación del conjunto de datos. Si se produce un error de servicio, intente volver a llamar a la API más tarde. Si se produce un error de cliente, compruebe los parámetros de entrada de la llamada a la API del conjunto de datos que ha fallado.

Type: Cadena

Valores válidos: SUCCESS | SERVICE\_ERROR | CLIENT\_ERROR

Obligatorio: No

## Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- [AWS SDK para C++](#)
- [AWS SDK para Go](#)
- [AWSSDK para Java V2](#)
- [SDK de AWS para Ruby V3](#)

## DatasetSource

El origen que utiliza Amazon Rekognition Custom Labels para crear un conjunto de datos. Para utilizar un archivo de manifiesto de formato de Amazon Sagemaker, especifique la ubicación del bucket de S3 en el `GroundTruthManifest`. El bucket de S3; debe estar en su cuenta de AWS. Para crear una copia de un conjunto de datos existente, especifique el nombre de recurso de Amazon (ARN) de un conjunto de datos existente en `DatasetArn`.

Debe especificar un valor para `DatasetArn` o `GroundTruthManifest`, pero no ambos. Si proporciona ambos valores o si no especifica ningún valor, se produce una excepción `InvalidParameterException`.

Para obtener más información, consulte [CreateDataset \(p. 525\)](#).

## Contenido

### DatasetArn

El ARN de un conjunto de datos de etiquetas personalizadas de Amazon Rekognition que desea copiar.

Type: Cadena

Restricciones de longitud: Longitud mínima de 20. La longitud máxima es de 2048 caracteres.

Patrón: (^arn:[a-z\d-]+:rekognition:[a-z\d-]+\:\d{12}:project\/[a-zA-Z0-9\_.\-\-]{1,255}\/dataset\/(train|test)\/[0-9]+\$)

Obligatorio: No

### GroundTruthManifest

El bucket de S3 que contiene un archivo de manifiesto de formato de Amazon Sagemaker Ground Truth.

Tipo: objeto [GroundTruthManifest \(p. 782\)](#)

Obligatorio: No

## Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- [AWS SDK para C++](#)
- [AWS SDK para Go](#)
- [AWSSDK para Java V2](#)
- [SDK de AWS para Ruby V3](#)

## DatasetStats

Proporciona estadísticas sobre un conjunto de datos. Para obtener más información, consulte [DescribeDataset \(p. 559\)](#).

### Contenido

#### ErrorEntries

El número total de entradas que contienen al menos un error.

Type: Entero

Rango válido: Valor mínimo de 0.

Obligatorio: No

#### LabeledEntries

El número total de imágenes del conjunto de datos que tienen etiquetas.

Type: Entero

Rango válido: Valor mínimo de 0.

Obligatorio: No

#### TotalEntries

El número total de imágenes en el conjunto de datos.

Type: Entero

Rango válido: Valor mínimo de 0.

Obligatorio: No

#### TotalLabels

El número total de etiquetas declaradas en el conjunto de datos.

Type: Entero

Rango válido: Valor mínimo de 0.

Obligatorio: No

## Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- [AWS SDK para C++](#)
- [AWS SDK para Go](#)
- [AWSSDK para Java V2](#)
- [SDK de AWS para Ruby V3](#)

## DetectionFilter

Conjunto de parámetros que le permiten filtrar determinados resultados de los resultados devueltos.

### Contenido

#### 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.

Type: Float

Rango válido: Valor mínimo de 0. Valor máximo de 1.

Obligatorio: No

#### MinBoundingBoxWidth

Establece el ancho mínimo del cuadro delimitador de palabras. Las palabras con anchos de cuadros delimitadores inferiores a este valor se excluyen del resultado. El valor es relativo al ancho del fotograma de vídeo.

Type: Float

Rango válido: Valor mínimo de 0. Valor máximo de 1.

Obligatorio: No

#### MinConfidence

Establece la confianza de la detección de palabras. Las palabras con confianza de detección por debajo de esto se excluyen del resultado. Los valores deben estar entre 50 y 100, ya que el texto en vídeo no devolverá ningún resultado inferior a 50.

Type: Float

Rango válido: Valor mínimo de 0. Valor máximo de 100.

Obligatorio: No

### Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- [AWS SDK para C++](#)
- [AWS SDK para Go](#)
- [AWSSDK para Java V2](#)
- [SDK de AWS para Ruby V3](#)

## DetectTextFilters

Conjunto de parámetros opcionales que puede utilizar para establecer los criterios que debe cumplir el texto para incluirlo en la respuesta.`WordFilter`analiza la altura, el ancho y la confianza mínima de una palabra.`RegionOfInterest`permite configurar una región específica de la imagen en la que buscar texto.

### Contenido

#### RegionsOfInterest

Filtro que se centra en un área determinada de la imagen. Usa un`BoundingBox`para establecer la región de la imagen.

Type: Matrices de`RegionOfInterest` (p. 816)objects

Miembros de matriz: El número mínimo es 0 elementos. Número máximo de 10 elementos.

Obligatorio: No

#### WordFilter

Conjunto de parámetros que le permiten filtrar determinados resultados de los resultados devueltos.

Tipo: objeto `DetectionFilter` (p. 763)

Obligatorio: No

### Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- [AWS SDK para C++](#)
- [AWS SDK para Go](#)
- [AWSSDK para Java V2](#)
- [SDK de AWS para Ruby V3](#)

## DistributeDataset

Un conjunto de datos de formación o un conjunto de datos de prueba utilizado en una operación de distribución de conjuntos de datos. Para obtener más información, consulte [DistributeDatasetEntries \(p. 600\)](#).

### Contenido

Arn

El nombre de recurso de Amazon (ARN) del conjunto de datos que desea utilizar.

Type: Cadena

Restricciones de longitud: Longitud mínima de 20. La longitud máxima es de 2048 caracteres.

Patrón: (^arn:[a-zA-Z\d-]+:rekognition:[a-zA-Z\d-]+\d{12}:project\/[a-zA-Z0-9\_.\-\\_]{1,255}\dataset\/(train|test)\/[0-9]+\$)

Obligatorio: Sí

### Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- [AWS SDK para C++](#)
- [AWS SDK para Go](#)
- [AWSSDK para Java V2](#)
- [SDK de AWS para Ruby V3](#)

## Emotion

Las emociones que parecen expresarse en la cara y el nivel de confianza en la determinación. La API solo está determinando la apariencia física del rostro de una persona. No es una determinación del estado emocional interno y no debe utilizarse de esa manera. Por ejemplo, una persona que finja estar triste con el rostro podría no estar triste emocionalmente.

### Contenido

#### Confidence

Nivel de confianza en la determinación.

Type: Float

Rango válido: Valor mínimo de 0. Valor máximo de 100.

Obligatorio: No

#### Type

Tipo de emoción detectada.

Type: Cadena

Valores válidos: HAPPY | SAD | ANGRY | CONFUSED | DISGUSTED | SURPRISED | CALM | UNKNOWN | FEAR

Obligatorio: No

### Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- [AWS SDK para C++](#)
- [AWS SDK para Go](#)
- [AWSSDK para Java V2](#)
- [SDK de AWS para Ruby V3](#)

## EquipmentDetection

Información sobre un elemento de equipo de protección personal (EPP) detectado por [DetectProtectiveEquipment \(p. 592\)](#). Para obtener más información, consulte [DetectProtectiveEquipment \(p. 592\)](#).

### Contenido

#### BoundingBox

Cuadro delimitador que rodea el elemento del EPP detectado.

Tipo: objeto [BoundingBox \(p. 740\)](#)

Obligatorio: No

#### Confidence

La confianza de que Amazon Rekognition tiene que el cuadro delimitador ([BoundingBox](#)) contiene un elemento de EPP.

Type: Float

Rango válido: Valor mínimo de 0. Valor máximo de 100.

Obligatorio: No

#### CoversBodyPart

Información sobre la parte del cuerpo cubierta por el EPP detectado.

Tipo: objeto [CoversBodyPart \(p. 752\)](#)

Obligatorio: No

#### Type

Tipo de EPP detectado.

Type: Cadena

Valores válidos: FACE\_COVER | HAND\_COVER | HEAD\_COVER

Obligatorio: No

## Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- [AWS SDK para C++](#)
- [AWS SDK para Go](#)
- [AWSSDK para Java V2](#)
- [SDK de AWS para Ruby V3](#)

## EvaluationResult

Los resultados de la evaluación para la formación de un modelo.

### Contenido

#### F1Score

Puntuación de F1 para la evaluación de todas las etiquetas. La métrica de puntuación de F1 evalúa la precisión general y el rendimiento de recuperación del modelo como un único valor. Un valor superior indica una mayor precisión y rendimiento de retirada. Una puntuación más baja indica que la precisión, la retirada o ambos están funcionando mal.

Type: Float

Obligatorio: No

#### Summary

El bucket de S3 que contiene el resumen de entrenamiento.

Tipo: objeto [Summary](#) (p. 832)

Obligatorio: No

### Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- [AWS SDK para C++](#)
- [AWS SDK para Go](#)
- [AWSSDK para Java V2](#)
- [SDK de AWS para Ruby V3](#)

## Eyeglasses

Indica si la cara lleva anteojos o no y el nivel de confianza en la determinación.

### Contenido

#### Confidence

Nivel de confianza en la determinación.

Type: Float

Rango válido: Valor mínimo de 0. Valor máximo de 100.

Obligatorio: No

#### Value

: un valor booleano que indica si el rostro lleva anteojos o no.

Type: Booleano

Obligatorio: No

### Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- [AWS SDK para C++](#)
- [AWS SDK para Go](#)
- [AWSSDK para Java V2](#)
- [SDK de AWS para Ruby V3](#)

## EyeOpen

Indica si los ojos de la cara están abiertos o no y el nivel de confianza en la determinación.

### Contenido

#### Confidence

Nivel de confianza en la determinación.

Type: Float

Rango válido: Valor mínimo de 0. Valor máximo de 100.

Obligatorio: No

#### Value

: un valor booleano que indica si los ojos en la cara están abiertos.

Type: Booleano

Obligatorio: No

### Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- [AWS SDK para C++](#)
- [AWS SDK para Go](#)
- [AWSSDK para Java V2](#)
- [SDK de AWS para Ruby V3](#)

## Face

Describe las propiedades de la cara, como el cuadro delimitador, el ID de cara, el ID de imagen de la imagen de entrada y el ID de imagen externo que ha asignado.

### Contenido

#### BoundingBox

Caja delimitadora de la cara.

Tipo: objeto [BoundingBox](#) (p. 740)

Obligatorio: No

#### Confidence

Nivel de confianza de que el cuadro delimitador contiene una cara (y no un objeto diferente, como un árbol).

Type: Float

Rango válido: Valor mínimo de 0. Valor máximo de 100.

Obligatorio: No

#### ExternalImageId

Identificador que asigna a todas las rostros en la imagen de entrada.

Type: Cadena

Limitaciones de longitud: Longitud mínima de 1. La longitud máxima es de 255 caracteres.

Patrón: [ a-zA-Z0-9\_.\-\:]<sup>+</sup>

Obligatorio: No

#### FaceId

Identificador único que Amazon Rekognition asigna a la cara.

Type: Cadena

Patrón: [ 0-9a-f ]{8}-[ 0-9a-f ]{4}-[ 0-9a-f ]{4}-[ 0-9a-f ]{4}-[ 0-9a-f ]{12}

Obligatorio: No

#### ImageId

Identificador único que Amazon Rekognition asigna a la imagen de entrada.

Type: Cadena

Patrón: [ 0-9a-f ]{8}-[ 0-9a-f ]{4}-[ 0-9a-f ]{4}-[ 0-9a-f ]{4}-[ 0-9a-f ]{12}

Obligatorio: No

#### IndexFacesModelVersion

Versión del modelo de detección y almacenamiento de rostros que se utilizó al indexar el vector de cara.

Type: Cadena

Patrón: [ 0-9\.\. ]+

Obligatorio: No

## Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- [AWS SDK para C++](#)
- [AWS SDK para Go](#)
- [AWSSDK para Java V2](#)
- [SDK de AWS para Ruby V3](#)

## FaceDetail

Estructura que contiene atributos de la cara que detectó el algoritmo.

UNA `FaceDetail` contiene los atributos faciales predeterminados o todos los atributos faciales. Los atributos predeterminados son `BoundingBox`, `Confidence`, `Landmarks`, `Pose`, y `Quality`.

[GetFaceDetection \(p. 615\)](#) es la única operación de vídeo almacenado de Amazon Rekognition Video que puede devolver un `FaceDetail` objeto con todos los atributos. Para especificar qué atributos se deben devolver, utilice la `FaceAttributes` parámetro de entrada para [StartFaceDetection \(p. 693\)](#). Las siguientes operaciones de Amazon Rekognition Video devuelven solo los atributos predeterminados. Las operaciones de inicio correspondientes no tienen un `FaceAttributes` parámetro de entrada.

- [GetCelebrityRecognition](#)
- [GetPersonTracking](#)
- [GetFaceSearch](#)

La imagen de Amazon Rekognition [DetectFaces \(p. 578\)](#) y [IndexFaces \(p. 644\)](#) las operaciones pueden devolver todos los atributos faciales. Para especificar qué atributos se deben devolver, utilice la `Attributes` parámetro de entrada para `DetectFaces`. Para `IndexFaces`, utilice el `DetectAttributes` parámetro de entrada.

## Contenido

### AgeRange

Rango de edades estimado, en años, para la cara. La baja representa la edad estimada más baja y la alta representa la edad estimada más alta.

Tipo: objeto [AgeRange \(p. 735\)](#)

Obligatorio: No

### Beard

Indica si la cara tiene barba o no y el nivel de confianza en la determinación.

Tipo: objeto [Beard \(p. 738\)](#)

Obligatorio: No

### BoundingBox

Caja delimitadora de la cara. Atributo predeterminado.

Tipo: objeto [BoundingBox \(p. 740\)](#)

Obligatorio: No

### Confidence

Nivel de confianza de que el cuadro delimitador contiene una cara (y no un objeto diferente, como un árbol). Atributo predeterminado.

Type: Float

Rango válido: Valor mínimo de 0. Valor máximo de 100.

Obligatorio: No

### Emotions

Las emociones que parecen expresarse en la cara y el nivel de confianza en la determinación. La API solo está determinando la apariencia física del rostro de una persona. No es una determinación del estado emocional interno y no debe utilizarse de esa manera. Por ejemplo, una persona que finja estar triste con el rostro podría no estar triste emocionalmente.

Tipo: Matriz de[Emotion \(p. 766\)](#)objects

Obligatorio: No

### Eyeglasses

Indica si la cara lleva anteojos o no y el nivel de confianza en la determinación.

Tipo: objeto [Eyeglasses \(p. 769\)](#)

Obligatorio: No

### EyesOpen

Indica si los ojos de la cara están abiertos o no y el nivel de confianza en la determinación.

Tipo: objeto [EyeOpen \(p. 770\)](#)

Obligatorio: No

### Gender

El género previsto de un rostro detectado.

Tipo: objeto [Gender \(p. 780\)](#)

Obligatorio: No

### Landmarks

Indica la ubicación de los puntos de referencia en la cara. Atributo predeterminado.

Tipo: Matriz de[Landmark \(p. 795\)](#)objects

Obligatorio: No

### MouthOpen

Indica si la boca de la cara está abierta o no y el nivel de confianza en la determinación.

Tipo: objeto [MouthOpen \(p. 797\)](#)

Obligatorio: No

### Mustache

Indica si la cara tiene bigote o no y el nivel de confianza en la determinación.

Tipo: objeto [Mustache \(p. 798\)](#)

Obligatorio: No

### Pose

Indica la postura del rostro tal como determina su cabeceo, balanceo y desviación. Atributo predeterminado.

Tipo: objeto [Pose \(p. 806\)](#)

Obligatorio: No

#### Quality

Identifica el brillo y la nitidez de la imagen. Atributo predeterminado.

Tipo: objeto [ImageQuality \(p. 788\)](#)

Obligatorio: No

#### Smile

Indica si la cara sonríe o no y el nivel de confianza en la determinación.

Tipo: objeto [Smile \(p. 823\)](#)

Obligatorio: No

#### Sunglasses

Indica si la cara lleva gafas de sol o no y el nivel de confianza en la determinación.

Tipo: objeto [Sunglasses \(p. 833\)](#)

Obligatorio: No

## Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- [AWS SDK para C++](#)
- [AWS SDK para Go](#)
- [AWSSDK para Java V2](#)
- [SDK de AWS para Ruby V3](#)

## FaceDetection

Información acerca de un rostro detectado en una solicitud de análisis de vídeo y el momento en que el rostro se detectó en el vídeo.

### Contenido

#### Face

Propiedades de cara de la cara detectada.

Tipo: objeto [FaceDetail](#) (p. 773)

Obligatorio: No

#### Timestamp

Tiempo, en milisegundos desde el principio del vídeo, cuando se detectó el rostro.

Type: Long

Obligatorio: No

### Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- [AWS SDK para C++](#)
- [AWS SDK para Go](#)
- [AWSSDK para Java V2](#)
- [SDK de AWS para Ruby V3](#)

## FaceMatch

Proporciona metadatos faciales. Además, también proporciona la confianza en la coincidencia de esta cara con la cara de entrada.

### Contenido

#### Face

Describe las propiedades de la cara como el cuadro delimitador, el ID de cara, el ID de imagen de la imagen de origen y el ID de imagen externo que ha asignado.

Tipo: objeto [Face](#) (p. 771)

Obligatorio: No

#### Similarity

Confianza en la concordancia de esta cara con la que los rostros rostros

Type: Float

Rango válido: Valor mínimo de 0. Valor máximo de 100.

Obligatorio: No

### Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- [AWS SDK para C++](#)
- [AWS SDK para Go](#)
- [AWSSDK para Java V2](#)
- [SDK de AWS para Ruby V3](#)

## FaceRecord

Objeto que contiene los metadatos de la cara (almacenados en la base de datos backend) y atributos faciales que se detectan pero no se almacenan en la base de datos.

### Contenido

#### Face

Describe las propiedades de la cara, como el cuadro delimitador, el ID de cara, el ID de imagen de la imagen de entrada y el ID de imagen externo que ha asignado.

Tipo: objeto [Face](#) (p. 771)

Obligatorio: No

#### FaceDetail

Estructura que contiene atributos de la cara que detectó el algoritmo.

Tipo: objeto [FaceDetail](#) (p. 773)

Obligatorio: No

### Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- [AWS SDK para C++](#)
- [AWS SDK para Go](#)
- [AWSSDK para Java V2](#)
- [SDK de AWS para Ruby V3](#)

## FaceSearchSettings

Parámetros de reconocimiento facial de entrada para un procesador de transmisión de Amazon Rekognition. `FaceRecognitionSettings` es un parámetro de solicitud para [CreateStreamProcessor \(p. 537\)](#).

### Contenido

#### CollectionId

Id. de una colección que contiene caras que desea buscar.

Type: Cadena

Limitaciones de longitud: Longitud mínima de 1. La longitud máxima es de 255 caracteres.

Patrón: [a-zA-Z0-9\_.\-\+]+

Obligatorio: No

#### FaceMatchThreshold

Puntuación de confianza mínima de coincidencia facial que debe cumplirse para devolver un resultado para una cara reconocida. El valor predeterminado es 80. 0 es la confianza más baja. 100 es la confianza más alta. Se aceptan valores entre 0 y 100 y los valores inferiores a 80 se establecen en 80.

Type: Float

Rango válido: Valor mínimo de 0. Valor máximo de 100.

Obligatorio: No

### Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- [AWS SDK para C++](#)
- [AWS SDK para Go](#)
- [AWSSDK para Java V2](#)
- [SDK de AWS para Ruby V3](#)

## Gender

El género previsto de un rostro detectado.

Amazon Rekognition realiza predicciones binarias del sexo (masculino/femenino) basadas en la apariencia física de un rostro de una imagen determinada. Este tipo de predicción no está diseñado para clasificar la identidad de género de una persona y no debe utilizarse Amazon Rekognition para realizar esta determinación. Por ejemplo, un actor masculino que lleva una peluca de pelo largo y pendientes para un papel podría predecirse como mujer.

El uso de Amazon Rekognition para hacer predicciones binarias de género es más adecuado para casos de uso en los que las estadísticas agregadas de distribución de género deben analizarse sin identificar a usuarios específicos. Por ejemplo, el porcentaje de usuarios femeninos en comparación con usuarios masculinos 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.

## Contenido

Confidence

Nivel de confianza en la predicción.

Type: Float

Rango válido: Valor mínimo de 0. Valor máximo de 100.

Obligatorio: No

Value

El género previsto de la cara.

Type: Cadena

Valores válidos: Male | Female

Obligatorio: No

## Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- [AWS SDK para C++](#)
- [AWS SDK para Go](#)
- [AWSSDK para Java V2](#)
- [SDK de AWS para Ruby V3](#)

## Geometry

Información sobre dónde se encuentra un objeto ([DetectCustomLabels \(p. 573\)](#)) o texto ([DetectText \(p. 596\)](#)) se encuentra en una imagen.

### Contenido

#### BoundingBox

Una representación gruesa alineada con el eje de la ubicación del elemento detectado en la imagen.

Tipo: objeto [BoundingBox \(p. 740\)](#)

Obligatorio: No

#### Polygon

Dentro del cuadro delimitador, un polígono de grano fino alrededor del elemento detectado.

Type: Matriz de[Point \(p. 805\)](#)objects

Obligatorio: No

### Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- [AWS SDK para C++](#)
- [AWS SDK para Go](#)
- [AWSSDK para Java V2](#)
- [SDK de AWS para Ruby V3](#)

## GroundTruthManifest

El bucket de S3 que contiene un archivo de manifiesto de formato de Amazon Sagemaker Ground Truth.

### Contenido

#### S3Object

Proporciona el nombre de bucket de S3 y el nombre de objeto.

La región del bucket de S3 que contiene el objeto S3 debe coincidir con la región que utiliza para las operaciones de Amazon Rekognition.

Para que Amazon Rekognition procese un objeto S3, el usuario debe tener permiso para acceder al objeto de S3. Para obtener más información, consulte [Políticas de Amazon Rekognition basadas en recursos \(p. 487\)](#).

Tipo: objeto [S3Object \(p. 817\)](#)

Obligatorio: No

### Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- [AWS SDK para C++](#)
- [AWS SDK para Go](#)
- [AWSSDK para Java V2](#)
- [SDK de AWS para Ruby V3](#)

## HumanLoopActivationOutput

Muestra los resultados de la evaluación humana en bucle. Si no hay HumanLoopArn, la entrada no activó la revisión humana.

### Contenido

#### HumanLoopActivationConditionsEvaluationResults

Muestra el resultado de las evaluaciones de afecciones, incluidas las condiciones que activaron una revisión humana.

Type: Cadena

Restricciones de longitud: La longitud máxima es de 10240 caracteres.

: obligatorio No

#### HumanLoopActivationReasons

Muestra si se necesitaba una revisión humana y por qué.

Type: Matriz de cadenas

Miembros de la matriz: Número mínimo de 1 elemento.

: obligatorio No

#### HumanLoopArn

Nombre de recurso de Amazon (ARN) del HumanLoop creado.

Type: Cadena

Restricciones de longitud: La longitud máxima es de 256 caracteres.

: obligatorio No

### Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- [AWS SDK para C++](#)
- [AWS SDK para Go](#)
- [AWSSDK para Java V2](#)
- [SDK de AWS para Ruby V3](#)

## HumanLoopConfig

Configura la definición de flujo a la que se enviará la imagen si se cumple una de las condiciones. También puedes establecer ciertos atributos de la imagen antes de revisar.

### Contenido

#### DataAttributes

Establece los atributos de los datos de entrada.

Tipo: objeto [HumanLoopDataAttributes \(p. 785\)](#)

Obligatorio: No

#### FlowDefinitionArn

El nombre de recurso de Amazon (ARN) de la definición del flujo. Puede crear una definición de flujo utilizando Amazon Sagemaker.[CreateFlowDefinitionOperación](#).

Type: Cadena

Restricciones de longitud: La longitud máxima es de 256 caracteres.

Obligatorio: Sí

#### HumanLoopName

El nombre de la revisión humana utilizada para esta imagen. Esto debe ser único dentro de una región.

Type: Cadena

Restricciones de longitud: Longitud mínima de 1. La longitud máxima es de 63 caracteres.

Patrón: ^[a-zA-Z0-9](-\*[a-zA-Z0-9])\*

Obligatorio: Sí

### Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- [AWS SDK para C++](#)
- [AWS SDK para Go](#)
- [AWSSDK para Java V2](#)
- [SDK de AWS para Ruby V3](#)

## HumanLoopDataAttributes

Permite definir los atributos de la imagen. En la actualidad, puede declarar una imagen libre de información de identificación personal.

### Contenido

#### ContentClassifiers

Establece si la imagen de entrada está libre de información de identificación personal.

Type: Matriz de cadenas

Miembros de matrices: Número máximo de 256 elementos.

Valores válidos: `FreeOfPersonallyIdentifiableInformation` | `FreeOfAdultContent`

Obligatorio: No

### Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- [AWS SDK para C++](#)
- [AWS SDK para Go](#)
- [AWSSDK para Java V2](#)
- [SDK de AWS para Ruby V3](#)

## Image

Proporciona la imagen de entrada en bytes o como objeto S3.

Puede transferir bytes de imágenes a una operación API de Amazon Rekognition utilizando laBytes propiedad. Por ejemplo, debería utilizar elBytes para transferir una imagen cargada desde un sistema de archivos local. Bytes de imagen pasados mediante elBytes La propiedad debe estar codificada con base64. Es posible que el código no necesite codificar bytes de imagen si utiliza un SDK de AWS para llamar a las operaciones de la API de Amazon Rekognition.

Para obtener más información, consulte [Análisis de una imagen cargada desde un sistema de archivos local \(p. 39\)](#).

Puede transferir imágenes almacenadas en un bucket de S3 a una operación API de Amazon Rekognition utilizando laS3Object propiedad. Las imágenes almacenadas en un bucket de S3 no necesitan codificarse en base64.

La región del bucket de S3 que contiene el objeto S3 debe coincidir con la región que utiliza para las operaciones de Amazon Rekognition.

Si utiliza la CLI de AWS para llamar a las operaciones de Amazon Rekognition, no es posible pasar bytes de imágenes utilizando la propiedad Bytes. Debe cargar primero la imagen en un bucket de Amazon S3 y, a continuación, llamar a la operación utilizando la propiedad S3Object.

Para que Amazon Rekognition procese un objeto de S3, el usuario debe tener permiso para acceder al objeto de S3. Para obtener más información, consulte [Políticas de Amazon Rekognition basadas en recursos \(p. 487\)](#).

## Contenido

### Bytes

Blob de bytes de imagen de hasta 5 MBs.

Type: Objeto de datos binarios codificados en Base64

Restricciones de longitud: Longitud mínima de 1. La longitud máxima es de 5242880 caracteres.

Obligatorio: No

### S3Object

Identifica un objeto de S3 como origen de imagen.

Tipo: objeto [S3Object \(p. 817\)](#)

Obligatorio: No

## Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- [AWS SDK para C++](#)
- [AWS SDK para Go](#)
- [AWSSDK para Java V2](#)
- [SDK de AWS para Ruby V3](#)



## ImageQuality

Identifica el brillo y la nitidez de la imagen del rostro.

### Contenido

#### Brightness

Valor que representa el brillo de la cara. El servicio devuelve un valor comprendido entre 0 y 100 (ambos incluidos). Un valor más elevado implica una imagen facial más brillante.

Type: Float

Obligatorio: No

#### Sharpness

Valor que representa la nitidez del rostro. El servicio devuelve un valor comprendido entre 0 y 100 (ambos incluidos). Un valor más elevado implica una imagen de cara más nítida.

Type: Float

Obligatorio: No

### Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- [AWS SDK para C++](#)
- [AWS SDK para Go](#)
- [AWSSDK para Java V2](#)
- [SDK de AWS para Ruby V3](#)

## Instance

Instancia de una etiqueta devuelta por Amazon Rekognition Image ([DetectLabels \(p. 583\)](#)) o por Amazon Rekognition Video ([GetLabelDetection \(p. 626\)](#)).

### Contenido

#### BoundingBox

Posición de la instancia de etiqueta en la imagen.

Tipo: objeto [BoundingBox \(p. 740\)](#)

Obligatorio: No

#### Confidence

La confianza que Amazon Rekognition tiene en la precisión del cuadro delimitador.

Type: Float

Rango válido: Valor mínimo de 0. Valor máximo de 100.

Obligatorio: No

### Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- [AWS SDK para C++](#)
- [AWS SDK para Go](#)
- [AWSSDK para Java V2](#)
- [SDK de AWS para Ruby V3](#)

## KinesisDataStream

El flujo de datos de Kinesis Amazon Rekognition al que se transmiten los resultados del análisis de un procesador de secuencias de Amazon Rekognition. Para obtener más información, consulte [CreateStreamProcessor \(p. 537\)](#).

### Contenido

Arn

ARN de la transmisión de Amazon Kinesis Data Streams de salida.

Type: Cadena

Patrón: (^arn:([a-z\d-]+):kinesis:([a-z\d-]+):\d{12}:+\$)

Obligatorio: No

### Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- [AWS SDK para C++](#)
- [AWS SDK para Go](#)
- [AWSSDK para Java V2](#)
- [SDK de AWS para Ruby V3](#)

## KinesisVideoStream

Transmisión de transmisión de vídeo de Kinesis que proporciona la transmisión de vídeo de origen para un procesador de streaming de Amazon Rekognition Video. Para obtener más información, consulte [CreateStreamProcessor \(p. 537\)](#).

### Contenido

Arn

ARN del flujo de vídeo de Kinesis que transmite el vídeo de origen en.

Type: Cadena

Patrón: (^arn:([a-z\d-]+):kinesisvideo:([a-z\d-]+):\d{12}:+\$)

Obligatorio: No

### Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- [AWS SDK para C++](#)
- [AWS SDK para Go](#)
- [AWSSDK para Java V2](#)
- [SDK de AWS para Ruby V3](#)

## KnownGender

La identidad de género conocida de la celebridad que coincide con la identificación proporcionada. La identidad de género conocida puede ser hombre, mujer, no binaria o no incluida en la lista.

### Contenido

#### Type

Un valor de cadena de la información de género conocida sobre la celebridad.

Type: Cadena

Valores válidos: `Male` | `Female` | `Nonbinary` | `Unlisted`

Obligatorio: No

### Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- [AWS SDK para C++](#)
- [AWS SDK para Go](#)
- [AWSSDK para Java V2](#)
- [SDK de AWS para Ruby V3](#)

## Label

Estructura que contiene detalles sobre la etiqueta detectada, incluidos el nombre, las instancias detectadas, las etiquetas principales y el nivel de confianza.

### Contenido

#### Confidence

Nivel de confianza.

Type: Float

Rango válido: Valor mínimo de 0. Valor máximo de 100.

Obligatorio: No

#### Instances

Si `Label` representa un objeto, `Instances` contiene los cuadros delimitadores de cada instancia del objeto detectado. Se devuelven cajas delimitadoras de etiquetas de objetos comunes, como personas, automóviles, muebles, prendas de vestir o mascotas.

Type: Matriz de [Instance \(p. 789\)](#) objects

Obligatorio: No

#### Name

Nombre (etiqueta) del objeto o escena.

Type: Cadena

Obligatorio: No

#### Parents

Etiquetas principales de una etiqueta. La respuesta incluye todas las etiquetas de antepasados.

Type: Matriz de [Parent \(p. 801\)](#) objects

Obligatorio: No

## Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- [AWS SDK para C++](#)
- [AWS SDK para Go](#)
- [AWSSDK para Java V2](#)
- [SDK de AWS para Ruby V3](#)

## LabelDetection

Información sobre una etiqueta detectada en una solicitud de análisis de vídeo y la hora en que se detectó la etiqueta en el vídeo.

### Contenido

#### Label

Detalles sobre la etiqueta detectada.

Tipo: objeto [Label](#) (p. 793)

Obligatorio: No

#### Timestamp

Tiempo, en milisegundos desde el principio del vídeo, cuando se detectó la etiqueta.

Type: Long

Obligatorio: No

### Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- [AWS SDK para C++](#)
- [AWS SDK para Go](#)
- [AWSSDK para Java V2](#)
- [SDK de AWS para Ruby V3](#)

## Landmark

Indica la ubicación del hito en la cara.

### Contenido

#### Type

Tipo de hito.

Type: Cadena

Valores válidos: eyeLeft | eyeRight | nose | mouthLeft | mouthRight | leftEyeBrowLeft | leftEyeBrowRight | leftEyeBrowUp | rightEyeBrowLeft | rightEyeBrowRight | rightEyeBrowUp | leftEyeLeft | leftEyeRight | leftEyeUp | leftEyeDown | rightEyeLeft | rightEyeRight | rightEyeUp | rightEyeDown | noseLeft | noseRight | mouthUp | mouthDown | leftPupil | rightPupil | upperJawlineLeft | midJawlineLeft | chinBottom | midJawlineRight | upperJawlineRight

Obligatorio: No

X

La coordenada x del hito expresada como una proporción del ancho de la imagen. La coordenada x se mide desde el lado izquierdo de la imagen. Por ejemplo, si la imagen tiene 700 píxeles de ancho y la coordenada x del hito es de 350 píxeles, este valor es 0,5.

Type: Float

Obligatorio: No

Y

La coordenada y del hito expresada como una proporción de la altura de la imagen. La coordenada y se mide desde la parte superior de la imagen. Por ejemplo, si la altura de la imagen es de 200 píxeles y la coordenada y del punto de referencia es de 50 píxeles, este valor es 0,25.

Type: Float

Obligatorio: No

### Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- [AWS SDK para C++](#)
- [AWS SDK para Go](#)
- [AWSSDK para Java V2](#)
- [SDK de AWS para Ruby V3](#)

## ModerationLabel

Proporciona información sobre un solo tipo de contenido inapropiado, no deseado u ofensivo que se encuentra en una imagen o vídeo. Cada tipo de contenido moderado tiene una etiqueta dentro de una taxonomía jerárquica. Para obtener una lista de etiquetas de moderación en Amazon Rekognition, consulte[Uso de las API de moderación de imagen y vídeo](#). Para obtener más información, consulte [Moderación de contenido \(p. 298\)](#).

### Contenido

#### Confidence

Especifica la confianza de Amazon Rekognition de que la etiqueta se ha identificado correctamente.

Si no especifica el `MinConfidence` parámetro en la llamada `aDetectModerationLabels`, la operación devuelve etiquetas con un valor de confianza superior o igual al 50 por ciento.

Type: Float

Rango válido: Valor mínimo de 0. Valor máximo de 100.

Obligatorio: No

#### Name

El nombre de la etiqueta del tipo de contenido no seguro detectado en la imagen.

Type: Cadena

Obligatorio: No

#### ParentName

Nombre de la etiqueta padre. Las etiquetas del nivel superior de la jerarquía tienen la etiqueta principal".

Type: Cadena

Obligatorio: No

## Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- [AWS SDK para C++](#)
- [AWS SDK para Go](#)
- [AWSSDK para Java V2](#)
- [SDK de AWS para Ruby V3](#)

## MouthOpen

Indica si la boca de la cara está abierta o no y el nivel de confianza en la determinación.

### Contenido

#### Confidence

Nivel de confianza en la determinación.

Type: Float

Rango válido: Valor mínimo de 0. Valor máximo de 100.

Obligatorio: No

#### Value

: un valor booleano que indica si la boca de la cara está abierta o no.

Type: Booleano

Obligatorio: No

### Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- [AWS SDK para C++](#)
- [AWS SDK para Go](#)
- [AWSSDK para Java V2](#)
- [SDK de AWS para Ruby V3](#)

## Mustache

Indica si la cara tiene bigote o no y el nivel de confianza en la determinación.

### Contenido

#### Confidence

Nivel de confianza en la determinación.

Type: Float

Rango válido: : un valor mínimo de 0. Valor máximo de 100.

Obligatorio: No

#### Value

: un valor booleano que indica si la cara tiene bigote o no.

Type: Booleano

Obligatorio: No

### Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- [AWS SDK para C++](#)
- [AWS SDK para Go](#)
- [AWSSDK para Java V2](#)
- [SDK de AWS para Ruby V3](#)

## NotificationChannel

El tema de Amazon Simple Notification Service en el que Amazon Rekognition publica el estado de realización de una operación de análisis de vídeo de. Para obtener más información, consulte [Llamar a las operaciones de Amazon Rekognition Video \(p. 66\)](#). Tenga en cuenta que el tema de Amazon SNS debe tener un nombre de tema que comience por AmazonRekognitionServiceRole para acceder al tema. Para obtener más información, consulte [Acceso a varios temas de Amazon SNS](#).

### Contenido

#### RoleArn

El ARN de un rol de IAM que otorga a Amazon Rekognition permisos de publicación en el tema de Amazon SNS.

Type: Cadena

Patrón: `arn:aws:iam::\d{12}:role/?[a-zA-Z_0-9+=,.@\-_/.]+`

Obligatorio: Sí

#### SNSTopicArn

Tema de Amazon SNS en el que Amazon Rekognition publica el estado de finalización.

Type: Cadena

Patrón: `(^arn:aws:sns:.*:\w{12}:+$)`

Obligatorio: Sí

### Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- [AWS SDK para C++](#)
- [AWS SDK para Go](#)
- [AWSSDK para Java V2](#)
- [SDK de AWS para Ruby V3](#)

## OutputConfig

Ubicación de bucket y carpeta de S3 en la que se coloca la salida del entrenamiento.

### Contenido

#### S3Bucket

Bucket de S3 en el que se coloca la salida de entrenamiento.

Type: Cadena

Restricciones de longitud: Longitud mínima de 3. La longitud máxima es de 255 caracteres.

Patrón: [ 0-9A-Za-z\.\-\_]\*

Obligatorio: No

#### S3KeyPrefix

El prefijo aplicado a los archivos de salida de esta formación.

Type: Cadena

Restricciones de longitud: La longitud máxima es de 1024 caracteres.

Obligatorio: No

### Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- [AWS SDK para C++](#)
- [AWS SDK para Go](#)
- [AWSSDK para Java V2](#)
- [SDK de AWS para Ruby V3](#)

## Parent

Etiqueta principal de una etiqueta. Una etiqueta puede tener 0, 1 o más padres.

### Contenido

#### Name

El nombre del elemento principal.

Type: Cadena

Obligatorio: No

### Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- [AWS SDK para C++](#)
- [AWS SDK para Go](#)
- [AWSSDK para Java V2](#)
- [SDK de AWS para Ruby V3](#)

## PersonDetail

Detalles sobre una persona detectada en una solicitud de análisis de vídeo.

### Contenido

#### BoundingBox

Cuadro delimitador alrededor de la persona detectada.

Tipo: objeto [BoundingBox \(p. 740\)](#)

Obligatorio: No

#### Face

Detalles de la cara de la persona detectada.

Tipo: objeto [FaceDetail \(p. 773\)](#)

Obligatorio: No

#### Index

Identificador de la persona detectada en un vídeo. Se utiliza para realizar un seguimiento de la persona a lo largo del vídeo. Amazon Rekognition no almacena el identificador.

Type: Long

Obligatorio: No

### Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- [AWS SDK para C++](#)
- [AWS SDK para Go](#)
- [AWSSDK para Java V2](#)
- [SDK de AWS para Ruby V3](#)

## PersonDetection

Detalles e información de seguimiento de rutas durante una sola vez que se realiza el seguimiento de la ruta de una persona en un vídeo. Las operaciones de Amazon Rekognition que rastrean las rutas de las personas devuelven una serie de PersonDetection objetos con elementos para cada vez que se realiza un seguimiento de la ruta de una persona en un vídeo.

Para obtener más información, consulte [GetPersonTracking \(p. 630\)](#).

## Contenido

### Person

Detalles sobre una persona cuyo camino se ha rastreado en un vídeo.

Tipo: objeto [PersonDetail \(p. 802\)](#)

Obligatorio: No

### Timestamp

es el tiempo, en milisegundos desde el principio del vídeo, cuando se rastreó el camino de la persona.

Type: Long

Obligatorio: No

## Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- [AWS SDK para C++](#)
- [AWS SDK para Go](#)
- [AWSSDK para Java V2](#)
- [SDK de AWS para Ruby V3](#)

## PersonMatch

Información sobre una persona cuyo rostro coincide con una (s) cara (s) de una colección de Amazon Rekognition. Incluye información sobre los rostros de la colección Amazon Rekognition ([FaceMatch \(p. 777\)](#)), información sobre la persona ([PersonDetail \(p. 802\)](#)) y el sello de tiempo para cuando se detectó a la persona en un vídeo. Una matriz de PersonMatch los objetos son devueltos por [GetFaceSearch \(p. 620\)](#).

### Contenido

#### FaceMatches

Información sobre los rostros de la colección de entrada que coinciden con el rostro de una persona en el vídeo.

Type: Matriz de [FaceMatch \(p. 777\)](#) objects

Obligatorio: No

#### Person

Información sobre la persona emparejada.

Tipo: objeto [PersonDetail \(p. 802\)](#)

Obligatorio: No

#### Timestamp

El tiempo, en milisegundos desde el comienzo del video, durante el que se emparejó a la persona en el video.

Type: Long

Obligatorio: No

## Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- [AWS SDK para C++](#)
- [AWS SDK para Go](#)
- [AWSSDK para Java V2](#)
- [SDK de AWS para Ruby V3](#)

## Point

Las coordenadas X e Y de un punto de una imagen. Los valores X e Y devueltos son ratio del tamaño de imagen global. Por ejemplo, si la imagen de entrada es 700x200 y la operación devuelve X=0,5 e Y=0,25, el punto se encuentra en la coordenada de píxeles (350,50) de la imagen.

Una matriz de `Point` objetos, `Polygon`, es devuelto por [DetectText \(p. 596\)](#) y por [DetectCustomLabels \(p. 573\)](#). `Polygon` representa un polígono de grano fino alrededor de un elemento detectado. Para obtener más información, consulte [Geometry \(p. 781\)](#).

### Contenido

X

El valor de la coordenada X de un punto de un `Polygon`.

Type: Float

Obligatorio: No

Y

El valor de la coordenada Y de un punto de un `Polygon`.

Type: Float

Obligatorio: No

### Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- [AWS SDK para C++](#)
- [AWS SDK para Go](#)
- [AWSSDK para Java V2](#)
- [SDK de AWS para Ruby V3](#)

## Pose

Indica la postura del rostro tal como determina su cabeceo, balanceo y desviación.

### Contenido

#### Pitch

Valor que representa la rotación de la cara en el eje de inclinación.

Type: Float

Rango válido: Valor mínimo de -180. Valor máximo de 180.

Obligatorio: No

#### Roll

Valor que representa la rotación de la cara en el eje de balanceo.

Type: Float

Rango válido: Valor mínimo de -180. Valor máximo de 180.

Obligatorio: No

#### Yaw

Valor que representa la rotación de cara en el eje de guiñada.

Type: Float

Rango válido: Valor mínimo de -180. Valor máximo de 180.

Obligatorio: No

## Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- [AWS SDK para C++](#)
- [AWS SDK para Go](#)
- [AWSSDK para Java V2](#)
- [SDK de AWS para Ruby V3](#)

## ProjectDescription

Una descripción de un proyecto de etiquetas personalizadas de Amazon Rekognition. Para obtener más información, consulte [DescribeProjects \(p. 561\)](#).

### Contenido

#### CreationTimestamp

La marca temporal de Unix para la fecha y hora en que se creó el proyecto.

Type: Marca temporal

Obligatorio: No

#### Datasets

Información sobre los conjuntos de datos de capacitación y prueba del proyecto.

Type: Matriz de[DatasetMetadata \(p. 759\)](#)objects

Obligatorio: No

#### ProjectArn

El nombre de recurso de Amazon (ARN) del proyecto.

Type: Cadena

Restricciones de longitud: Longitud mínima de 20. La longitud máxima es de 2048 caracteres.

Patrón: (^arn:[a-zA-Z\d-]+:rekognition:[a-zA-Z\d-]+\d{12}:project\/[a-zA-Z0-9\_.\-\]{1,255}\/[0-9]+\$)

Obligatorio: No

#### Status

El estado actual del proyecto.

Type: Cadena

Valores válidos: CREATING | CREATED | DELETING

Obligatorio: No

## Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- [AWS SDK para C++](#)
- [AWS SDK para Go](#)
- [AWSSDK para Java V2](#)
- [SDK de AWS para Ruby V3](#)

## ProjectVersionDescription

Descripción de una versión de un modelo de etiquetas personalizadas de Amazon Rekognition.

### Contenido

#### BillableTrainingTimeInSeconds

La duración, en segundos, que se le facturó por un entrenamiento exitoso de la versión del modelo. Este valor solo se devuelve si la versión del modelo se ha entrenado correctamente.

Type: Long

Rango válido: Valor mínimo de 0.

Obligatorio: No

#### CreationTimestamp

Fecha y hora de Unix para la fecha y hora en que comenzó la formación.

Type: Marca temporal

Obligatorio: No

#### EvaluationResult

Los resultados de la formación. `EvaluationResults` solo se devuelve si el entrenamiento tiene éxito.

Tipo: objeto [EvaluationResult \(p. 768\)](#)

Obligatorio: No

#### KmsKeyId

Identificador de la clave de AWS Key Management Service (clave de AWS KMS) que se utilizó para cifrar el modelo durante la formación.

Type: Cadena

Restricciones de longitud: Longitud mínima de 1. La longitud máxima es de 2048 caracteres.

Patrón: ^[A-Za-z0-9][A-Za-z0-9:\_/+=@.-]{0,2048}\$

Obligatorio: No

#### ManifestSummary

Ubicación del manifiesto de resumen. El manifiesto de resumen proporciona resultados de validación de datos agregados para los conjuntos de datos de formación y prueba.

Tipo: objeto [GroundTruthManifest \(p. 782\)](#)

Obligatorio: No

#### MinInferenceUnits

El número mínimo de unidades de inferencia usadas por el modelo. Para obtener más información, consulte [StartProjectVersion \(p. 709\)](#).

Type: Entero

Rango válido: Valor mínimo de 1.

Obligatorio: No

OutputConfig

Ubicación en la que se guardan los resultados de la formación.

Tipo: objeto [OutputConfig \(p. 800\)](#)

Obligatorio: No

ProjectVersionArn

Nombre de recurso de Amazon (ARN) de la versión del modelo.

Type: Cadena

Restricciones de longitud: Longitud mínima de 20. La longitud máxima es de 2048 caracteres.

Patrón: (^arn:[a-zA-Z\d-]+:rekognition:[a-zA-Z\d-]+\:\d{12}:project\/[a-zA-Z0-9\_.\-\]{1,255}\version\/[a-zA-Z0-9\_.\-\]{1,255}\/[0-9]+\\$)

Obligatorio: No

Status

El estado actual de la versión del modelo.

Type: Cadena

Valores válidos: TRAINING\_IN\_PROGRESS | TRAINING\_COMPLETED | TRAINING\_FAILED | STARTING | RUNNING | FAILED | STOPPING | STOPPED | DELETING

Obligatorio: No

StatusMessage

Mensaje descriptivo de un error o advertencia que se ha producido.

Type: Cadena

Obligatorio: No

TestingDataResult

Contiene información sobre los resultados de las pruebas.

Tipo: objeto [TestingDataResult \(p. 836\)](#)

Obligatorio: No

TrainingDataResult

Contiene información sobre los resultados del entrenamiento.

Tipo: objeto [TrainingDataResult \(p. 841\)](#)

Obligatorio: No

TrainingEndTimestamp

Fecha y hora de Unix en que finalizó la formación del modelo.

Type: Marca temporal

Obligatorio: No

## Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- [AWS SDK para C++](#)
- [AWS SDK para Go](#)
- [AWSSDK para Java V2](#)
- [SDK de AWS para Ruby V3](#)

## ProtectiveEquipmentBodyPart

Información sobre una pieza del cuerpo detectada por [DetectProtectiveEquipment \(p. 592\)](#) que contiene EPP. Una matriz de ProtectiveEquipmentBodyPart los objetos se devuelven para cada persona detectada por [DetectProtectiveEquipment](#).

### Contenido

#### Confidence

La confianza que Amazon Rekognition tiene en la precisión de detección de la parte del cuerpo detectada.

Type: Float

Rango válido: Valor mínimo de 0. Valor máximo de 100.

Obligatorio: No

#### EquipmentDetections

Una serie de artículos de equipo de protección personal detectados alrededor de una parte del cuerpo.

Type: Matriz de [EquipmentDetection \(p. 767\)](#) objects

Obligatorio: No

#### Name

La parte del cuerpo detectada.

Type: Cadena

Valores válidos: FACE | HEAD | LEFT\_HAND | RIGHT\_HAND

Obligatorio: No

## Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- [AWS SDK para C++](#)
- [AWS SDK para Go](#)
- [AWSSDK para Java V2](#)
- [SDK de AWS para Ruby V3](#)

## ProtectiveEquipmentPerson

Una persona detectada por una llamada a [DetectProtectiveEquipment \(p. 592\)](#). La API devuelve a todas las personas detectadas en la imagen de entrada en una matriz de ProtectiveEquipmentPerson objetos.

### Contenido

#### BodyParts

Conjunto de partes del cuerpo detectadas en el cuerpo de una persona (incluidas las partes del cuerpo sin EPP).

Type: Matriz de [ProtectiveEquipmentBodyPart \(p. 811\)](#) objects

Obligatorio: No

#### BoundingBox

Un cuadro delimitador alrededor de la persona detectada.

Tipo: objeto [BoundingBox \(p. 740\)](#)

Obligatorio: No

#### Confidence

Confianza de Amazon Rekognition de que el cuadro delimitador contiene a una persona.

Type: Float

Rango válido: Valor mínimo de 0. Valor máximo de 100.

Obligatorio: No

#### Id

Identificador de la persona detectada. El identificador solo es exclusivo para una sola llamada [aDetectProtectiveEquipment](#).

Type: Entero

Rango válido: Valor mínimo de 0.

Obligatorio: No

## Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- [AWS SDK para C++](#)
- [AWS SDK para Go](#)
- [AWSSDK para Java V2](#)
- [SDK de AWS para Ruby V3](#)

## ProtectiveEquipmentSummarizationAttributes

Especifica los atributos de resumen que se deben devolver de una llamada [aDetectProtectiveEquipment \(p. 592\)](#). Puede especificar qué tipos de EPP desea resumir. También puede especificar un valor de confianza mínimo para las detecciones. La información resumida se devuelve en [elSummary\(ProtectiveEquipmentSummary \(p. 814\)\)](#) de la respuesta de `aDetectProtectiveEquipment`. El resumen incluye qué personas en una imagen se detectaron que llevaban los tipos solicitados de equipo de protección de personas (EPP), qué personas se detectaron que no llevaban EPP y las personas en las que no se pudo determinar. Para obtener más información, consulte [ProtectiveEquipmentSummary \(p. 814\)](#).

### Contenido

#### MinConfidence

El nivel de confianza mínimo para el que desea obtener información resumida. El nivel de confianza se aplica a la detección de personas, la detección de partes del cuerpo, la detección de equipos y la cobertura de partes del cuerpo. Amazon Rekognition no devuelve información de resumen con confianza que este valor especificado. No hay un valor predeterminado.

Especificar un `MinConfidence` valor que está entre el 50-100% como `detectProtectiveEquipment` devuelve predicciones solo 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%.

Type: Float

Rango válido: Valor mínimo de 0. Valor máximo de 100.

Obligatorio: Sí

#### RequiredEquipmentTypes

Una serie de tipos de equipos de protección personal para los que desea obtener información resumida. Si se detecta que una persona lleva un tipo de equipo obligatorio, el documento de identidad de la persona se añade a la `PersonsWithRequiredEquipment` campo array devuelto en [ProtectiveEquipmentSummary \(p. 814\)](#) por `DetectProtectiveEquipment`.

Type: Matriz de cadenas

Valores válidos: `FACE_COVER` | `HAND_COVER` | `HEAD_COVER`

Obligatorio: Sí

### Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- [AWS SDK para C++](#)
- [AWS SDK para Go](#)
- [AWSSDK para Java V2](#)
- [SDK de AWS para Ruby V3](#)

## ProtectiveEquipmentSummary

Información resumida de los artículos requeridos de equipo de protección personal (EPP) detectados en personas mediante una llamada a [DetectProtectiveEquipment \(p. 592\)](#). Especifique el tipo de EPP requerido en el `SummarizationAttributes` ([ProtectiveEquipmentSummarizationAttributes \(p. 813\)](#)) parámetro de entrada. El resumen incluye qué personas se detectaron que llevaban el equipo de protección personal requerido (`PersonsWithRequiredEquipment`), qué personas se detectaron que no llevaban el EPP requerido (`PersonsWithoutRequiredEquipment`) y las personas en las que no se pudo tomar una determinación (`PersonsIndeterminate`).

Para obtener un total de cada categoría, utilice el tamaño de la matriz de campos. Por ejemplo, para averiguar cuántas personas han detectado que usan el EPP especificado, utilice el tamaño del `PersonsWithRequiredEquipment` matriz. Si desea obtener más información acerca de una persona, como la ubicación ([BoundingBox \(p. 740\)](#)) de la persona de la imagen, utilice el ID de persona en cada elemento de matriz. Cada ID de persona coincide con el campo ID de un [ProtectiveEquipmentPerson \(p. 812\)](#) objeto devuelto en el `Persons` matriz por `DetectProtectiveEquipment`.

### Contenido

#### PersonsIndeterminate

Una serie de identificaciones para personas en las que no fue posible determinar si llevaban equipo de protección personal.

Type: Matriz de números enteros

Rango válido: Valor mínimo de 0.

obligatorio: obligatorio No

#### PersonsWithoutRequiredEquipment

Una serie de identificaciones para personas que no llevan todos los tipos de EPP especificados en el `RequiredEquipmentTypes` del equipo de protección personal detectado.

Type: Matriz de números enteros

Rango válido: Valor mínimo de 0.

obligatorio: obligatorio No

#### PersonsWithRequiredEquipment

Una serie de identificaciones para personas que usan equipo de protección personal detectado.

Type: Matriz de números enteros

Rango válido: Valor mínimo de 0.

obligatorio: obligatorio No

### Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- [AWS SDK para C++](#)
- [AWS SDK para Go](#)

- [AWSSDK para Java V2](#)
- [SDK de AWS para Ruby V3](#)

## RegionOfInterest

Especifica una ubicación dentro del marco que Rekognition busca texto. Usa un `BoundingBox` objeto para establecer una región de la pantalla.

Se incluye una palabra en la región si la palabra está más de la mitad en esa región. Si hay más de una región, la palabra se comparará con todas las regiones de la pantalla. Cualquier palabra más de la mitad de una región se conserva en los resultados.

### Contenido

`BoundingBox`

El cuadro que representa una región de interés en pantalla.

Tipo: objeto [BoundingBox](#) (p. 740)

Obligatorio: No

### Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- [AWS SDK para C++](#)
- [AWS SDK para Go](#)
- [AWSSDK para Java V2](#)
- [SDK de AWS para Ruby V3](#)

## S3Object

Proporciona el nombre de bucket de S3 y el nombre de objeto.

La región del bucket de S3 que contiene el objeto S3 debe coincidir con la región que utiliza para las operaciones de Amazon Rekognition.

Para que Amazon Rekognition procese un objeto S3, el usuario debe tener permiso para acceder al objeto S3. Para obtener más información, consulte [Políticas de Amazon Rekognition basadas en recursos \(p. 487\)](#).

### Contenido

#### Bucket

Nombre del bucket de S3.

Type: Cadena

Restricciones de longitud: Longitud mínima de 3. La longitud máxima es de 255 caracteres.

Patrón: [ 0-9A-Za-z\.\-\_]\*

Obligatorio: No

#### Name

Nombre de la clave de objeto de S3.

Type: Cadena

Restricciones de longitud: Longitud mínima de 1. La longitud máxima es de 1024 caracteres.

Obligatorio: No

#### Version

Si el bucket está habilitado, puede especificar la versión del objeto.

Type: Cadena

Restricciones de longitud: Longitud mínima de 1. La longitud máxima es de 1024 caracteres.

Obligatorio: No

## Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- [AWS SDK para C++](#)
- [AWS SDK para Go](#)
- [AWSSDK para Java V2](#)
- [SDK de AWS para Ruby V3](#)

## SegmentDetection

Segmento técnico de detección de tomas detectadas en un vídeo. Una matriz de SegmentDetection los objetos que contienen todos los segmentos detectados en un vídeo almacenado son devueltos por [GetSegmentDetection \(p. 635\)](#).

### Contenido

#### DurationFrames

Duración de un segmento de vídeo, expresada en fotogramas.

Type: Long

Rango válido: Valor mínimo de 0.

Obligatorio: No

#### DurationMillis

La duración del segmento detectado en milisegundos.

Type: Long

Rango válido: Valor mínimo de 0.

Obligatorio: No

#### DurationSMpte

Duración del código de tiempo del segmento detectado en formato SMPTE.

Type: Cadena

Obligatorio: No

#### EndFrameNumber

El número de fotograma al final de un segmento de vídeo, utilizando un índice de fotogramas que comienza por 0.

Type: Long

Rango válido: Valor mínimo de 0.

Obligatorio: No

#### EndTimecodeSMPTE

Código de tiempo SMPTE con precisión de fotogramas, desde el inicio de un vídeo, para el final de un segmento detectado. EndTimecode está en HH:mm:SS: FF formato (y; ff para velocidades de fotogramas de caída).

Type: Cadena

Obligatorio: No

#### EndTimestampMillis

La hora de finalización del segmento detectado, en milisegundos, desde el inicio del vídeo. Este valor se redondea hacia abajo.

Type: Long

Obligatorio: No

**ShotSegment**

Si el segmento es una detección de disparo, contiene información sobre la detección de disparos.

Tipo: objeto [ShotSegment \(p. 822\)](#)

Obligatorio: No

**StartFrameNumber**

El número de fotograma del inicio de un segmento de vídeo, utilizando un índice de fotogramas que comienza por 0.

Type: Long

Rango válido: Valor mínimo de 0.

Obligatorio: No

**StartTimecodeSMPTE**

Código de tiempo SMPTE con precisión de fotogramas, desde el inicio de un vídeo, para el inicio de un segmento detectado. `StartTimecode` está en HH:mm:SS: FR formato (y; fr para velocidades de fotogramas de caída).

Type: Cadena

Obligatorio: No

**StartTimestampMillis**

La hora de inicio del segmento detectado en milisegundos desde el comienzo del vídeo. Este valor se redondea hacia abajo. Por ejemplo, si la marca de hora real es de 100,6667 milisegundos, Amazon Rekognition Video devuelve un valor de 100 millis.

Type: Long

Obligatorio: No

**TechnicalCueSegment**

Si el segmento es una señal técnica, contiene información sobre la señal técnica.

Tipo: objeto [TechnicalCueSegment \(p. 834\)](#)

Obligatorio: No

**Type**

Tipo de segmento. Los valores válidos son TECHNICAL\_CUE y SHOT.

Type: Cadena

Valores válidos: TECHNICAL\_CUE | SHOT

Obligatorio: No

## Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- [AWS SDK para C++](#)

- [AWS SDK para Go](#)
- [AWSSDK para Java V2](#)
- [SDK de AWS para Ruby V3](#)

## SegmentTypeInfo

Información sobre el tipo de segmento solicitado en una llamada a [StartSegmentDetection \(p. 712\)](#). Una matriz de SegmentTypeInfo los objetos se devuelven mediante la respuesta de [GetSegmentDetection \(p. 635\)](#).

### Contenido

#### ModelVersion

Versión del modelo que se utiliza para detectar segmentos.

Type: Cadena

Obligatorio: No

#### Type

Tipo de segmento (detección técnica de señal o disparo).

Type: Cadena

Valores válidos: TECHNICAL\_CUE | SHOT

Obligatorio: No

### Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- [AWS SDK para C++](#)
- [AWS SDK para Go](#)
- [AWSSDK para Java V2](#)
- [SDK de AWS para Ruby V3](#)

## ShotSegment

Información sobre un segmento de detección de tomas detectado en un vídeo. Para obtener más información, consulte [SegmentDetection \(p. 818\)](#).

### Contenido

#### Confidence

La confianza que Amazon Rekognition Video tiene en la precisión del segmento detectado.

Type: Float

Rango válido: Valor mínimo de 50. Valor máximo de 100.

Obligatorio: No

#### Index

Identificador para un segmento de detección de tomas detectado en un vídeo.

Type: Long

Rango válido: Valor mínimo de 0.

Obligatorio: No

### Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- [AWS SDK para C++](#)
- [AWS SDK para Go](#)
- [AWSSDK para Java V2](#)
- [SDK de AWS para Ruby V3](#)

## Smile

Indica si la cara sonríe o no y el nivel de confianza en la determinación.

### Contenido

#### Confidence

Nivel de confianza en la determinación.

Type: Float

Rango válido: Valor mínimo de 0. Valor máximo de 100.

Obligatorio: No

#### Value

: un valor booleano que indica si la cara sonríe o no.

Type: Booleano

Obligatorio: No

### Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- [AWS SDK para C++](#)
- [AWS SDK para Go](#)
- [AWSSDK para Java V2](#)
- [SDK de AWS para Ruby V3](#)

## StartSegmentDetectionFilters

Filtros aplicados a los segmentos técnicos de detección de tomas o tomas. Para obtener más información, consulte [StartSegmentDetection \(p. 712\)](#).

### Contenido

#### ShotFilter

Filtros específicos de las detecciones de disparos.

Tipo: objeto [StartShotDetectionFilter \(p. 825\)](#)

Obligatorio: No

#### TechnicalCueFilter

Filtros específicos de las señales técnicas.

Tipo: objeto [StartTechnicalCueDetectionFilter \(p. 826\)](#)

Obligatorio: No

### Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- [AWS SDK para C++](#)
- [AWS SDK para Go](#)
- [AWSSDK para Java V2](#)
- [SDK de AWS para Ruby V3](#)

## StartShotDetectionFilter

Filtros para los segmentos de detección de disparos devueltos por `GetSegmentDetection`. Para obtener más información, consulte [StartSegmentDetectionFilters \(p. 824\)](#).

### Contenido

#### MinSegmentConfidence

Especifica la confianza mínima que debe tener Amazon Rekognition Video para devolver un segmento detectado. La confianza representa cuán determinado Amazon Rekognition se identifica correctamente un segmento. 0 es la confianza más baja. 100 es la confianza más alta. Amazon Rekognition Video no devuelve ningún segmento con un nivel de confianza inferior al valor especificado.

Si no especifica `MinSegmentConfidence`, el `GetSegmentDetection` devuelve segmentos con valores de confianza mayores o iguales al 50 por ciento.

Type: Float

Rango válido: Valor mínimo de 50. Valor máximo de 100.

Obligatorio: No

### Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- [AWS SDK para C++](#)
- [AWS SDK para Go](#)
- [AWSSDK para Java V2](#)
- [SDK de AWS para Ruby V3](#)

## StartTechnicalCueDetectionFilter

Filtros para los segmentos técnicos devueltos por[GetSegmentDetection \(p. 635\)](#). Para obtener más información, consulte [StartSegmentDetectionFilters \(p. 824\)](#).

### Contenido

#### BlackFrame

Filtro que permite controlar la detección de fotogramas negros especificando los niveles de negro y la cobertura de píxeles de los píxeles negros de un fotograma. Los videos pueden provenir de varias fuentes, formatos y períodos de tiempo, con estándares diferentes y niveles de ruido variables para fotogramas negros que deben tenerse en cuenta.

Tipo: objeto [BlackFrame \(p. 739\)](#)

Obligatorio: No

#### MinSegmentConfidence

Especifica la confianza mínima que debe tener Amazon Rekognition Video para devolver un segmento detectado. La confianza representa cuán determinado Amazon Rekognition se identifica correctamente un segmento. 0 es la confianza más baja. 100 es la confianza más alta. Amazon Rekognition Video no devuelve ningún segmento con un nivel de confianza inferior al valor especificado.

Si no especifica `MinSegmentConfidence`, `GetSegmentDetection` devuelve segmentos con valores de confianza mayores o iguales al 50 por ciento.

Type: Float

Rango válido: Valor mínimo de 50. Valor máximo de 100.

Obligatorio: No

### Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- [AWS SDK para C++](#)
- [AWS SDK para Go](#)
- [AWSSDK para Java V2](#)
- [SDK de AWS para Ruby V3](#)

## StartTextDetectionFilters

Conjunto de parámetros opcionales que le permiten establecer los criterios que debe cumplir el texto para incluirse en la respuesta.`WordFilter`mira la altura, el ancho y la confianza mínima de una palabra.`RegionOfInterest`permite configurar una región específica de la pantalla para buscar texto.

### Contenido

#### RegionsOfInterest

Filtro centrado en un área determinada del marco. Usa un`BoundingBox`objeto para establecer la región de la pantalla.

Type: Matriz de[RegionOfInterest \(p. 816\)](#)`objects`

Miembros de matriz: El número mínimo es 0 elementos. Número máximo de 10 elementos.

Obligatorio: No

#### WordFilter

Filtros centrados en las cualidades del texto, como la confianza o el tamaño.

Tipo: objeto [DetectionFilter \(p. 763\)](#)

Obligatorio: No

### Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- [AWS SDK para C++](#)
- [AWS SDK para Go](#)
- [AWSSDK para Java V2](#)
- [SDK de AWS para Ruby V3](#)

## StreamProcessor

Objeto que reconoce rostros en un vídeo en streaming. Un procesador de flujo de Amazon Rekognition se crea mediante una llamada a [CreateStreamProcessor \(p. 537\)](#). Asignación de parámetros de solicitud para `CreateStreamProcessor` describir la fuente de transmisión de vídeo de Kinesis para la transmisión de vídeo, los parámetros de reconocimiento facial y dónde se deben transmitir los resultados del análisis.

### Contenido

#### Name

Nombre del procesador de transmisión de Amazon Rekognition.

Type: Cadena

Restricciones de longitud: Longitud mínima de 1. La longitud máxima es de 128 caracteres.

Patrón: [ a-zA-Z0-9\_.\-\+]<sup>+</sup>

Obligatorio: No

#### Status

El estado actual del procesador de transmisión de Amazon Rekognition.

Type: Cadena

Valores válidos: STOPPED | STARTING | RUNNING | FAILED | STOPPING

Obligatorio: No

### Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- [AWS SDK para C++](#)
- [AWS SDK para Go](#)
- [AWSSDK para Java V2](#)
- [SDK de AWS para Ruby V3](#)

## StreamProcessorInput

Información sobre el streaming de vídeo fuente.

### Contenido

#### KinesisVideoStream

El flujo de entrada de transmisión de vídeo de Kinesis para la transmisión de vídeo fuente.

Tipo: objeto [KinesisVideoStream \(p. 791\)](#)

Requerido: obligatorio No

### Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- [AWS SDK para C++](#)
- [AWS SDK para Go](#)
- [AWSSDK para Java V2](#)
- [SDK de AWS para Ruby V3](#)

## StreamProcessorOutput

Información sobre la transmisión de Amazon Kinesis Data Streams a la que un procesador de secuencias de Amazon Rekognition Video transmite los resultados de un análisis de vídeo. Para obtener más información, consulte [CreateStreamProcessor \(p. 537\)](#).

### Contenido

#### KinesisDataStream

Transmisión de Amazon Kinesis Data Streams a la que el procesador de secuencias de Amazon Rekognition transmite los resultados del análisis.

Tipo: objeto [KinesisDataStream \(p. 790\)](#)

Obligatorio: No

### Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- [AWS SDK para C++](#)
- [AWS SDK para Go](#)
- [AWSSDK para Java V2](#)
- [SDK de AWS para Ruby V3](#)

## StreamProcessorSettings

Parámetros de entrada utilizados para reconocer rostros en vídeo en streaming analizado por un procesador de streaming de Amazon Rekognition.

### Contenido

#### FaceSearch

Configuración de búsqueda facial para usar en una transmisión de vídeo.

Tipo: objeto [FaceSearchSettings \(p. 779\)](#)

Obligatorio: No

### Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- [AWS SDK para C++](#)
- [AWS SDK para Go](#)
- [AWSSDK para Java V2](#)
- [SDK de AWS para Ruby V3](#)

## Summary

El bucket de S3 que contiene el resumen de entrenamiento. El resumen de formación incluye métricas de evaluación agregadas para todo el conjunto de datos de pruebas y métricas de cada etiqueta individual.

Para obtener el resumen de formación, la ubicación del bucket de S3 llamando [DescribeProjectVersions \(p. 564\)](#).

## Contenido

### S3Object

Proporciona el nombre de bucket de S3 y el nombre de objeto.

La región del bucket de S3 que contiene el objeto S3 debe coincidir con la región que utiliza para las operaciones de Amazon Rekognition.

Para que Amazon Rekognition procese un objeto S3, el usuario debe tener permiso para acceder al objeto de S3. Para obtener más información, consulte [Políticas de Amazon Rekognition basadas en recursos \(p. 487\)](#).

Tipo: objeto [S3Object \(p. 817\)](#)

Obligatorio: No

## Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- [AWS SDK para C++](#)
- [AWS SDK para Go](#)
- [AWSSDK para Java V2](#)
- [SDK de AWS para Ruby V3](#)

## Sunglasses

Indica si la cara lleva gafas de sol o no y el nivel de confianza en la determinación.

### Contenido

#### Confidence

Nivel de confianza en la determinación.

Type: Float

Rango válido: Valor mínimo de 0. Valor máximo de 100.

Obligatorio: No

#### Value

Valor booleano que indica si el rostro lleva gafas de sol o no.

Type: Booleano

Obligatorio: No

### Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- [AWS SDK para C++](#)
- [AWS SDK para Go](#)
- [AWSSDK para Java V2](#)
- [SDK de AWS para Ruby V3](#)

## TechnicalCueSegment

Información sobre un segmento de referencia técnica. Para obtener más información, consulte [SegmentDetection \(p. 818\)](#).

### Contenido

#### Confidence

La confianza que Amazon Rekognition Video tiene en la precisión del segmento detectado.

Type: Float

Rango válido: Valor mínimo de 50. Valor máximo de 100.

Obligatorio: No

#### Type

Tipo de señal técnica.

Type: Cadena

Valores válidos: `ColorBars` | `EndCredits` | `BlackFrames` | `OpeningCredits` | `StudioLogo` | `Slate` | `Content`

Obligatorio: No

### Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- [AWS SDK para C++](#)
- [AWS SDK para Go](#)
- [AWSSDK para Java V2](#)
- [SDK de AWS para Ruby V3](#)

## TestingData

El conjunto de datos utilizado para las pruebas. Opcionalmente, siAutoCreate, Amazon Rekognition Custom Labels utiliza el conjunto de datos de formación para crear un conjunto de datos de prueba con una división temporal del conjunto de datos de formación.

### Contenido

#### Assets

Los activos utilizados para las pruebas.

Type: Matriz de[Asset \(p. 736\)](#)objects

Obligatorio: No

#### AutoCreate

Si se especifica, las etiquetas personalizadas de Amazon Rekognition dividen temporalmente el conjunto de datos de formación (80%) para crear un conjunto de datos de prueba (20%) para el trabajo de formación. Una vez finalizada la formación, el conjunto de datos de prueba no se almacena y el conjunto de datos de formación vuelve a su tamaño anterior.

Type: Booleano

Obligatorio: No

### Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- [AWS SDK para C++](#)
- [AWS SDK para Go](#)
- [AWSSDK para Java V2](#)
- [SDK de AWS para Ruby V3](#)

## TestingDataResult

Archivos de manifiesto de formato Sagemaker Groundtruth para los conjuntos de datos de entrada, salida y validación que se utilizan y crean durante las pruebas.

### Contenido

#### Input

El conjunto de datos de pruebas que se suministró para la formación.

Tipo: objeto [TestingData](#) (p. 835)

Obligatorio: No

#### Output

Subconjunto del conjunto de datos que se probó realmente. Es posible que algunas imágenes (activos) no se prueben debido al formato de los archivos y otros problemas.

Tipo: objeto [TestingData](#) (p. 835)

Obligatorio: No

#### Validation

Ubicación del manifiesto de validación de datos. El manifiesto de validación de datos se crea para el conjunto de datos de prueba durante la formación de modelos.

Tipo: objeto [ValidationData](#) (p. 843)

Obligatorio: No

### Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- [AWS SDK para C++](#)
- [AWS SDK para Go](#)
- [AWSSDK para Java V2](#)
- [SDK de AWS para Ruby V3](#)

## TextDetection

Información sobre una palabra o línea de texto detectada por [DetectText \(p. 596\)](#).

La `DetectedText` contiene el texto que Amazon Rekognition ha detectado en la imagen.

Cada palabra y línea tiene un identificador (`Id`). Cada palabra pertenece a una línea y tiene un identificador padre (`ParentId`) que identifica la línea de texto en la que aparece la palabra. La `palabraId` es también un índice de la palabra dentro de una línea de palabras.

Para obtener más información, consulte [Detección de texto \(p. 315\)](#).

## Contenido

### Confidence

La confianza que Amazon Rekognition tiene en la precisión del texto detectado y la precisión de la geometría que rodea el texto detectado.

Type: Float

Rango válido: Valor mínimo de 0. Valor máximo de 100.

Obligatorio: No

### DetectedText

La palabra o línea de texto reconocida por Amazon Rekognition.

Type: Cadena

Obligatorio: No

### Geometry

La ubicación del texto detectado en la imagen. Incluye un cuadro delimitador grueso alineado con ejes que rodea el texto y un polígono de grano más fino para obtener información espacial más precisa.

Tipo: objeto [Geometry \(p. 781\)](#)

Obligatorio: No

### Id

Identificador del texto detectado. El identificador solo es exclusivo para una sola llamada `aDetectText`.

Type: Entero

Rango válido: Valor mínimo de 0.

Obligatorio: No

### ParentId

Identificador principal del texto detectado identificado por el valor de `ID`. Si el tipo de texto detectado es `LINE`, el valor de `ParentId` es `null`.

Type: Entero

Rango válido: Valor mínimo de 0.

Obligatorio: No

Type

Tipo de texto que se ha detectado.

Type: Cadena

Valores válidos: LINE | WORD

Obligatorio: No

## Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- [AWS SDK para C++](#)
- [AWS SDK para Go](#)
- [AWSSDK para Java V2](#)
- [SDK de AWS para Ruby V3](#)

## TextDetectionResult

Información sobre el texto detectado en un vídeo. Incluye el texto detectado, el tiempo en milisegundos desde el inicio del vídeo en que se detectó el texto y dónde se detectó en la pantalla.

### Contenido

#### TextDetection

Detalles sobre el texto detectado en un vídeo.

Tipo: objeto [TextDetection \(p. 837\)](#)

Obligatorio: No

#### Timestamp

El tiempo, en milisegundos desde el comienzo del vídeo, cuando se detectó el texto.

Type: Long

Obligatorio: No

### Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- [AWS SDK para C++](#)
- [AWS SDK para Go](#)
- [AWSSDK para Java V2](#)
- [SDK de AWS para Ruby V3](#)

## TrainingData

El conjunto de datos utilizado para la formación.

### Contenido

#### Assets

Archivo de manifiesto de Sagemaker GroundTruth que contiene las imágenes de entrenamiento (activos).

Type: Matriz de[Asset \(p. 736\)](#)objects

Obligatorio: No

### Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- [AWS SDK para C++](#)
- [AWS SDK para Go](#)
- [AWSSDK para Java V2](#)
- [SDK de AWS para Ruby V3](#)

## TrainingDataResult

Archivos de manifiesto de formato Sagemaker Groundtruth para los conjuntos de datos de entrada, salida y validación que se utilizan y crean durante las pruebas.

### Contenido

#### Input

Los activos de formación que proporcionaste para la formación.

Tipo: objeto [TrainingData \(p. 840\)](#)

Obligatorio: No

#### Output

Las imágenes (activos) que han sido entrenados por Amazon Rekognition Custom Labels.

Tipo: objeto [TrainingData \(p. 840\)](#)

Obligatorio: No

#### Validation

Ubicación del manifiesto de validación de datos. El manifiesto de validación de datos se crea para el conjunto de datos de formación durante la formación de modelos.

Tipo: objeto [ValidationData \(p. 843\)](#)

Obligatorio: No

### Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- [AWS SDK para C++](#)
- [AWS SDK para Go](#)
- [AWSSDK para Java V2](#)
- [SDK de AWS para Ruby V3](#)

## UnindexedFace

Una cara que [IndexFaces](#) (p. 644) detectado, pero no indexado. Usar `Reasons` response para determinar por qué no se indexó una cara.

### Contenido

#### FaceDetail

Estructura que contiene atributos de una cara que [IndexFaces](#) detectado, pero no indexado.

Tipo: objeto [FaceDetail](#) (p. 773)

Obligatorio: No

#### Reasons

Una serie de razones que especifican por qué no se indexó una cara.

- EXTREME\_POSE - La cara se encuentra en una pose que no se puede detectar. Por ejemplo, la cabeza está demasiado alejada de la cámara.
- EXCEEDS\_MAX\_FACES: el número de caras detectadas ya es superior al especificado por el `MaxFaces` parámetro de entrada para [IndexFaces](#).
- LOW\_BRIGHTNESS - La imagen es demasiado oscura.
- LOW\_SHARPNESS - La imagen está demasiado borrosa.
- LOW\_CONFIDENCE - La cara se detectó con poca confianza.
- SMALL\_BOUNDING\_BOX - El cuadro delimitador alrededor de la cara es demasiado pequeño.

Type: Matriz de cadenas

Valores válidos: `EXCEEDS_MAX_FACES` | `EXTREME_POSE` | `LOW_BRIGHTNESS` |  
`LOW_SHARPNESS` | `LOW_CONFIDENCE` | `SMALL_BOUNDING_BOX` | `LOW_FACE_QUALITY`

Obligatorio: No

### Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- [AWS SDK para C++](#)
- [AWS SDK para Go](#)
- [AWSSDK para Java V2](#)
- [SDK de AWS para Ruby V3](#)

## ValidationData

Contiene la ubicación del bucket de Amazon S3 de los datos de validación de un trabajo de formación de modelos.

Los datos de validación incluyen información de error para líneas JSON individuales del conjunto de datos. Para obtener más información, consulte[Depuración de una formación de modelo fallida](#).

Obtendrá elValidationDataobjeto del conjunto de datos de formación ([TrainingDataResult \(p. 841\)](#)) y el conjunto de datos de prueba ([TestingDataResult \(p. 836\)](#)) llamando[DescribeProjectVersions \(p. 564\)](#).

La matriz de activos contiene un único[Asset \(p. 736\)](#)un objeto. La[GroundTruthManifest \(p. 782\)](#)del objeto Asset contiene la ubicación del bucket de S3 de los datos de validación.

## Contenido

### Assets

Los activos que componen los datos de validación.

Type: Matriz de[Asset \(p. 736\)](#)objects

Obligatorio: No

## Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- [AWS SDK para C++](#)
- [AWS SDK para Go](#)
- [AWSSDK para Java V2](#)
- [SDK de AWS para Ruby V3](#)

## Video

Archivo de vídeo almacenado en un bucket de Amazon S3. Operaciones de inicio de vídeo de Amazon Rekognition, tales como [StartLabelDetection \(p. 701\)](#) usar `Video` para especificar un vídeo para su análisis. Los formatos de archivo compatibles son .mp4, .mov y .avi.

### Contenido

#### S3Object

El nombre del bucket de Amazon S3 y el nombre del archivo del vídeo.

Tipo: objeto [S3Object \(p. 817\)](#)

Requiere: obligatorio No

### Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- [AWS SDK para C++](#)
- [AWS SDK para Go](#)
- [AWSSDK para Java V2](#)
- [SDK de AWS para Ruby V3](#)

## VideoMetadata

Información sobre un vídeo que Amazon Rekognition analizó. `videometadata` se devuelve en cada página de respuestas paginadas de una operación de vídeo de Amazon Rekognition.

### Contenido

#### Codec

Tipo de compresión utilizada en el vídeo analizado.

Type: Cadena

Obligatorio: No

#### ColorRange

Descripción del rango de valores de luminancia de un vídeo, LIMITADO (16 a 235) o FULL (0 a 255).

Type: Cadena

Valores válidos: `FULL` | `LIMITED`

Obligatorio: No

#### DurationMillis

Duración del vídeo en milisegundos.

Type: Long

Rango válido: Valor mínimo de 0.

Obligatorio: No

#### Format

Formato del vídeo analizado. Los valores posibles son MP4, MOV y AVI.

Type: Cadena

Obligatorio: No

#### FrameHeight

Dimensión vertical de píxeles del vídeo.

Type: Long

Rango válido: Valor mínimo de 0.

Obligatorio: No

#### FrameRate

Número de fotogramas por segundo en el vídeo.

Type: Float

Obligatorio: No

#### FrameWidth

Dimensión horizontal de píxeles del vídeo.

Type: Long

Rango válido: Valor mínimo de 0.

Obligatorio: No

## Véase también

Para obtener más información sobre el uso de esta API en un SDK de AWS de un lenguaje específico, consulte:

- [AWS SDK para C++](#)
- [AWS SDK para Go](#)
- [AWSSDK para Java V2](#)
- [SDK de AWS para Ruby V3](#)

# Directrices y cuotas de Amazon Rekognition

Las siguientes secciones indican directrices y cuotas de uso de Amazon Rekognition. Existen dos tipos de cuotas. Establecer cuotas como el tamaño máximo de la imagen, no pueden modificarse. Cuotas predeterminadas enumerados en el [AWS Cuotas de servicio](#) se puede cambiar siguiendo el procedimiento descrito en la [Cuotas predeterminadas \(p. 848\)](#) sección.

## Temas

- [Regiones admitidas \(p. 847\)](#)
- [Establecer cuotas \(p. 847\)](#)
- [Cuotas predeterminadas \(p. 848\)](#)

## Regiones admitidas

Para obtener una lista de AWS Regiones en las que Amazon Rekognition está disponible, consulte [Regiones y puntos de enlace de AWS](#) en la Referencia general de Amazon Web Services.

## Establecer cuotas

A continuación se ofrece una lista de los límites de Amazon Rekognition que no pueden modificarse. Para obtener información sobre los límites que puede cambiar, como los límites de transacciones por segundo (TPS), consulte [Cuotas predeterminadas \(p. 848\)](#).

Para ver los límites de etiquetas personalizadas de Amazon Rekognition, consulte [Directrices y cuotas de etiquetas personalizadas de Amazon Rekognition](#).

## Imagen de Amazon Rekognition

- El tamaño máximo de una imagen almacenada como un objeto de Amazon S3 se limita a 15 MB.
- Las dimensiones mínimas de la imagen son de 80 píxeles para el alto y el ancho. La dimensión mínima de imagen para `DetectProtectiveEquipment` es de 64 píxeles para el alto y el ancho.
- Dimensiones máximas de imagen para `DetectProtectiveEquipment` de 4096 píxeles para el ancho y el alto.
- Para ser detectado por `DetectProtectiveEquipment`, una persona debe ocupar más de 100 x 100 píxeles en una imagen de 800 x 1300. En el caso de las imágenes de más de 800 x 1300 píxeles, se necesitará un tamaño mínimo de persona proporcionalmente mayor.
- 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.
- 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 el `DetectProtectiveEquipment` API.

- 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 rostros que se pueden almacenar en una única colección de rostros es de 20 millones.
- El número máximo de rostros coincidentes devueltos por la API de búsqueda es de 4096.
- `DetectText` puede detectar hasta 100 palabras en una imagen.
- `DetectProtectiveEquipment` puede detectar equipos de protección personal 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, imágenes de entrada y vídeos \(p. 131\)](#).

## Vídeo almacenado de 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 simultáneos 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 analiza datos de audio solo admite códecs de audio ACC.
- 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`.

## Amazon Rekognition Video streaming de vídeo

- 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 transmisión de entrada de Kinesis Video y la transmisión de salida de Kinesis Data asociadas con un procesador de streaming de Amazon Rekognition Video no pueden compartirlos varios procesadores.
- Cualquier API de Amazon Rekognition Video que analiza datos de audio solo admite códecs de audio ACC.

## Cuotas predeterminadas

Puede consultar una lista de cuotas predeterminadas en [AWS Cuotas de servicio](#). Estos límites son predeterminados y se pueden cambiar. Para solicitar un aumento de límite, cree un caso. Para ver los límites de cuota actuales (valores de cuota aplicados), consulte [Cuotas de Amazon Rekognition Service](#). Para ver el historial de utilización de TPS para [API de imágenes de Amazon Rekognition](#), consulte la [Página Cuotas de Amazon Rekognition Service](#) y elige una operación de API específica para ver el historial de esa operación.

### Temas

- [Calcular el cambio de cuota de TPS \(p. 849\)](#)
- [Prácticas recomendadas para cuotas TPS \(p. 849\)](#)

- [Crear un caso para cambiar las cuotas TPS \(p. 849\)](#)

## Calcular el cambio de cuota de TPS

¿Cuál es el nuevo límite que estás solicitando? Las transacciones por segundo (TPS) son más relevantes en el pico de una carga de trabajo esperada. Es importante comprender el máximo de llamadas a API simultáneas en el pico de una carga de trabajo y el tiempo de respuesta (5 a 15 segundos). Tenga en cuenta que 5 segundos deberían ser el mínimo. A continuación se presentan dos ejemplos:

- Ejemplo 1: El máximo de usuarios simultáneos de autenticación facial (API de CompareFaces) que espero al comienzo de mi hora más ocupada es 1000. Estas respuestas se repartirán durante 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 máximo de llamadas simultáneas de detección de objetos (API DetectLabels) que se esperan al comienzo de mi hora más activa es de 250. Estas respuestas se repartirán durante 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 cuotas TPS

Las prácticas recomendadas para Transacciones por segundo (TPS) incluyen suavizar el tráfico puntiagudo, configurar reintentos y configurar el retroceso exponencial y la fluctuación.

1. Tráfico suave y puntiagudo. El tráfico puntiagudo afecta al rendimiento. Para obtener el máximo rendimiento de las transacciones asignadas por segundo (TPS), utilice una arquitectura sin servidor en cola u otro mecanismo para «suavizar» el tráfico de manera que sea más coherente. Para obtener ejemplos de código y referencias para el procesamiento de imágenes y vídeo a gran escala sin servidor con Rekognition, consulte [Procesamiento de imágenes y vídeo a gran escala con Amazon Rekognition](#).
2. Configurar reintentos. Sigue las directrices de [the section called “Control de errores” \(p. 122\)](#) para configurar los reintentos de los errores que les permiten.
3. Configure el retroceso exponencial y la fluctuación. La configuración del retroceso exponencial y la fluctuación a medida que configura los reintentos le permite mejorar el rendimiento alcanzable. Consulte [Reintentos de error y retardo exponencial en AWS](#).

## Crear un caso para cambiar las cuotas TPS

Para crear un caso, consulte [Crear caso](#) y responda a las siguientes preguntas:

- ¿Ha implementado el [the section called “Prácticas recomendadas para cuotas TPS” \(p. 849\)](#) para suavizar los picos de tráfico y configurar reintentos, retroceso exponencial y fluctuación?
- ¿Ha calculado el cambio de cuota de TPS que necesita? Si no, consulte [the section called “Calcular el cambio de cuota de TPS” \(p. 849\)](#).
- ¿Ha revisado su historial de uso de TPS para predecir con mayor precisión sus necesidades futuras? Para ver el historial de uso de TPS, consulte la [Página Cuotas de Amazon Rekognition Service](#).
- ¿Cuál es su caso de uso?
- ¿Qué API piensa usar?
- ¿En qué regiones piensa usar estas API?
- ¿Puede distribuir la carga en varias regiones?
- ¿Cuántas imágenes procesas diariamente?
- ¿Cuánto tiempo espera mantener este volumen (es un pico único o en curso)?

- ¿Cómo te bloquea el límite predeterminado? Revise la siguiente tabla de excepciones para confirmar el escenario que se encuentra.

Código de error	Excepción	Mensaje	¿Qué significa?	¿Se puede volver a intentar?
Código de estado HTTP 400	ProvisionedThroughputExceededException	Se ha excedido la tasa aprovisionada.	Indica la limitación controlada. Puede reintentar o evaluar una solicitud de aumento de límite.	Sí
Código de estado HTTP 400	ThrottlingException	Desaceleración; aumento repentino de la tasa de solicitudes.	Es posible que estés enviando tráfico punitivo y estrangulamiento. Debe dar forma al tráfico y hacerlo más fluido y coherente. A continuación, configure reintentos adicionales. Consulte prácticas recomendadas.	Sí
Código de estado HTTP 5xx	ThrottlingException (HTTP 500)	Service Unavailable	Indica que el backend se está ampliando para admitir la acción. Debe reintentar la solicitud.	Sí

Para obtener una comprensión detallada de los códigos de error, consulte [the section called “Control de errores” \(p. 122\)](#).

#### Note

Estos límites dependen de la región en la que te encuentres. El argumento para cambiar un límite afecta a la operación de API que solicitas, en la región en la que lo solicitas. Otras operaciones y regiones de API no se ven afectadas.

# Historial de documentos de Amazon Rekognition

En la siguiente tabla se describen los cambios importantes en cada versión delGuía para desarrolladores de Amazon Rekognition. Para obtener notificaciones sobre las actualizaciones de esta documentación, puede suscribirse a una fuente RSS.

- Última actualización de la documentación: 21 de mayo de 2021

update-history-change	update-history-description	update-history-date
<a href="#">Un nuevo nodo de la tabla de contenido muestra ejemplos de Amazon Rekognition alojados en GitHub (p. 851)</a>	Ejemplos de código actualizados delAWSEI repositorio de ejemplos de código aparece ahora en un nodo independiente en la guía para desarrolladores de Amazon Rekognition para facilitar el acceso.	22 de octubre de 2021
<a href="#">Amazon Rekognition puede detectar fotogramas negros y contenido principal del programa en segmentos de vídeo (p. 851)</a>	Amazon Rekognition puede identificar fotogramas negros, barras de colores, créditos de apertura, créditos finales, logotipos de estudio y contenido del programa principal como señales técnicas de un vídeo mediante elStartSegmentDetectionyGetSegmentDetectionoperaciones.	7 de junio de 2021
<a href="#">Amazon Rekognition DetectText puede detectar hasta 100 palabras en una imagen (p. 851)</a>	Puedes usar Amazon RekognitionDetectTextpara detectar hasta 100 palabras en una imagen.	21 de mayo de 2021
<a href="#">Amazon Rekognition ahora admite el etiquetado (p. 851)</a>	Ahora puedes utilizar etiquetas para identificar, organizar, buscar y filtrar colecciones de Amazon Rekognition, procesadores de streaming y modelos de etiquetas personalizadas.	25 de marzo de 2021
<a href="#">Amazon Rekognition ahora puede detectar equipos de protección personal (p. 851)</a>	Amazon Rekognition ahora puede detectar cubiertas de manos, carátulas y cubiertas para la cabeza de las personas de una imagen.	15 de octubre de 2020
<a href="#">Amazon Rekognition tiene nuevas categorías de moderación de contenido (p. 851)</a>	Las categorías de moderación de contenido de Amazon Rekognition ahora incluyen 6 categorías nuevas: Drogas, tabaco, alcohol, juegos de azar,	12 de octubre de 2020

<a href="#">Nuevo tutorial para mostrar los resultados de Amazon Rekognition Video de Kinesis Video Streams localmente (p. 851)</a>	gestos groseros y símbolos de odio.	20 de julio de 2020
<a href="#">Nuevo tutorial de Amazon Rekognition para usar Gstreamer (p. 851)</a>	Puede mostrar la salida de Amazon Rekognition Video desde un streaming de vídeo en Kinesis Video Streams en una fuente de vídeo local.	17 de julio de 2020
<a href="#">Amazon Rekognition ahora admite la segmentación de vídeos almacenados (p. 851)</a>	Con Gstreamer, puedes ingerir un vídeo en directo de una fuente de cámara de dispositivo a Amazon Rekognition Video a través de Kinesis Video Streams.	22 de junio de 2020
<a href="#">Amazon Rekognition ahora es compatible con las políticas de punto de enlace de la VPC de Amazon (p. 851)</a>	Con la API de segmentación de Amazon Rekognition Video asíncrona puede detectar fotogramas negros, barras de color, créditos finales y tomas en vídeos almacenados.	3 de marzo de 2020
<a href="#">Amazon Rekognition ahora admite la detección de texto en vídeos almacenados (p. 851)</a>	Al especificar una política, puede restringir el acceso a un punto de Amazon Rekognition Amazon VPC de Amazon.	17 de febrero de 2020
<a href="#">Amazon Rekognition ahora admite la Augmented AI (vista previa) y las etiquetas personalizadas de Amazon Rekognition (p. 851)</a>	Puede utilizar la API de Amazon Rekognition Video para detectar texto de forma asíncrona en un vídeo almacenado.	3 de diciembre de 2019
<a href="#">Amazon Rekognition es compatible con AWS PrivateLink (p. 851)</a>	Con las etiquetas personalizadas de Amazon Rekognition puede detectar objetos, escenas y conceptos especializados en imágenes creando su propio modelo de aprendizaje automático. DetectModerationLabels ya es compatible con Amazon Augmented AI (vista previa).	12 de septiembre de 2019
<a href="#">Filtrado de rostros de Amazon Rekognition (p. 851)</a>	Con AWS PrivateLink, puede establecer una conexión privada entre su VPC y Amazon Rekognition.	12 de septiembre de 2019
	Amazon Rekognition añade una compatibilidad mejorada con el filtrado de rostros a la operación de API de IndexFaces. Además, introduce el filtrado de rostros para las operaciones de API CompareFaces y SearchFacesByImage.	12 de septiembre de 2019

<a href="#">Actualización de ejemplos de Amazon Rekognition Video (p. 851)</a>	Se ha actualizado el código de ejemplo de Amazon Rekognition Video para crear y configurar el tema de Amazon SNS y la cola de Amazon SQS.	5 de septiembre de 2019
<a href="#">Se han añadido ejemplos de Ruby y Node.js (p. 851)</a>	Se han añadido ejemplos de Amazon Rekognition Image Ruby y Node.js para la detección síncrona de rostros y etiquetas.	19 de agosto de 2019
<a href="#">Se ha actualizado la detección de contenido no seguro (p. 851)</a>	La detección de contenido no seguro de Amazon Rekognition ahora puede detectar contenido violento.	9 de agosto de 2019
<a href="#">Actualización de las operaciones GetContentModeration (p. 851)</a>	GetContentModeration ahora devuelve la versión del modelo de detección de moderación que se utiliza para detectar contenido no seguro.	13 de febrero de 2019
<a href="#">Las operaciones GetLabelDetection y DetectModerationLabels se han actualizado (p. 851)</a>	Ahora, GetLabelDetection devuelve información del rectángulo delimitador de los objetos comunes y una taxonomía jerárquica de las etiquetas detectadas. También se devuelve la versión del modelo que se utiliza para detectar etiquetas. Además, DetectModerationLabels devuelve la versión del modelo que se utiliza para detectar contenido no seguro.	17 de enero de 2019
<a href="#">Actualización de las operaciones DetectFaces e IndexFaces (p. 851)</a>	Esta versión actualiza las operaciones DetectFaces e IndexFaces. Cuando el parámetro de entrada Attributes se establece en ALL, ahora hay cinco nuevas referencias de ubicación de rostros: upperJawlineLeft, midJawlineLeft, chinBottom, midJawlineRight y upperJawlineRight.	19 de noviembre de 2018
<a href="#">Actualización de la operación DetectLabels (p. 851)</a>	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

Se ha actualizado la operación IndexFaces (p. 851)	Con IndexFaces, ahora puede utilizar el parámetro de entrada QualityFilter para omitir los rostros detectados con baja calidad. También puede utilizar el parámetro de entrada MaxFaces 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
Se ha agregado la operación DescribeCollection (p. 851)	A partir de ahora, puede obtener información acerca de una colección existente llamando a la operación DescribeCollection.	22 de agosto de 2018
Nuevos ejemplos de Python (p. 851)	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 (p. 851)	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
es compatible con Amazon RekognitionAWS CloudTrail (p. 851)	Amazon Rekognition se integra con AWS CloudTrail, un servicio que proporciona un registro de las medidas adoptadas por un usuario, un rol o un AWSen Amazon Rekognition. Para obtener más información, consulte <a href="#">Registro de llamadas a la API de Amazon Rekognition con AWS CloudTrail</a> .	6 de abril de 2018
Analizar vídeos almacenados y en streaming. Nuevo índice (p. 851)	Para obtener información sobre el análisis de los vídeos almacenados, consulte <a href="#">Trabajar con vídeos almacenados</a> . Para obtener información sobre el análisis de los vídeos en streaming, consulte <a href="#">Trabajar con vídeos en streaming</a> . 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

Texto en modelos de detección de imágenes y rostros (p. 851)	Amazon Rekognition ahora puede detectar texto en imágenes. Para obtener más información, consulte <a href="#">Detección de texto</a> . Amazon Rekognition introduce el control de versiones para el modelo de aprendizaje profundo de detección de rostros. Para obtener más información, consulte <a href="#">Control de versiones del modelo</a> .	21 de noviembre de 2017
Reconocimiento de famosos (p. 851)	Ahora Amazon Rekognition puede analizar imágenes para detectar famosos. Para obtener más información, consulte <a href="#">Reconocimiento de famosos</a> .	8 de junio de 2017
Moderación de imágenes (p. 851)	Ahora Amazon Rekognition puede determinar si una imagen incluye contenido para adultos explícito o insinuante. Para obtener más información, consulte <a href="#">Detección de contenido no seguro</a> .	19 de abril de 2017
Rango de edad para las caras detectadas. Panel de métricas globales de Rekognition (p. 851)	Amazon Rekognition ahora devuelve una estimación del rango de edades, en años, para los rostros detectados por la API de Rekognition. Para obtener más información, consulte <a href="#">AgeRange</a> . La consola de Rekognition ahora tiene un panel de métricas que muestra gráficos de actividad para el total de métricas de Amazon CloudWatch para Rekognition durante un periodo de tiempo especificado. Para obtener más información, consulte <a href="#">Ejercicio 4: Consultar métricas totales (consola)</a> .	9 de febrero de 2017
Nuevo servicio y guía (p. 851)	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

# Glosario de AWS

Para ver la terminología más reciente de AWS, consulte el [Glosario de AWS](#) en la Referencia general de AWS.

Las traducciones son generadas a través de traducción automática. En caso de conflicto entre la traducción y la versión original de inglés, prevalecerá la versión en inglés.