
Amazon Rekognition

개발자 안내서



Amazon Rekognition: 개발자 안내서

Copyright © 2018 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon's trademarks and trade dress may not be used in connection with any product or service that is not Amazon's, in any manner that is likely to cause confusion among customers, or in any manner that disparages or discredits Amazon. All other trademarks not owned by Amazon are the property of their respective owners, who may or may not be affiliated with, connected to, or sponsored by Amazon.

Table of Contents

Amazon Rekognition이 무엇입니까?	1
Amazon Rekognition 및 HIPAA 자격	2
Amazon Rekognition을 처음 사용하십니까?	2
사용 방법	4
감지 및 인식 유형	4
레이블	4
얼굴	4
얼굴 검색	5
사람	5
유명 인사	5
텍스트 감지	5
안전하지 않은 콘텐츠	5
이미지 및 비디오 작업	5
Amazon Rekognition Image 작업	6
Amazon Rekognition Video 작업	6
비스토리지 및 스토리지 기반 작업	6
AWS SDK 또는 HTTP를 사용하여 Amazon Rekognition API 작업 호출	6
비스토리지 및 스토리지 API 작업	7
비스토리지 작업	7
스토리지 기반 API 작업	8
모델 버전 관리	9
시작하기	10
1단계: 계정 설정	10
AWS에 가입	10
IAM 사용자 생성	11
다음 단계	11
2단계: AWS CLI 및 AWS SDK 설정	11
다음 단계	13
3단계: AWS CLI 및 AWS SDK API를 사용하여 시작하기	13
AWS CLI 예제 형식	13
다음 단계	13
4단계: 콘솔 사용 시작하기	13
연습 1: 객체 및 장면 감지(콘솔)	14
연습 2: 얼굴 분석(콘솔)	18
연습 3: 얼굴 비교(콘솔)	21
연습 4: 집계 측정치 보기(콘솔)	23
인증 및 액세스 제어	24
인증	24
액세스 제어	25
액세스 관리 개요	25
Amazon Rekognition 리소스 및 작업	25
리소스 소유권 이해	26
리소스 액세스 관리	26
정책 요소 지정: 작업, 효과, 보안 주체	28
정책에서 조건 지정	28
자격 증명 기반 정책(IAM 정책) 사용	28
Amazon Rekognition 콘솔 사용에 필요한 권한	29
Amazon Rekognition에 대한 AWS 관리형(미리 정의된) 정책	29
고객 관리형 정책 예	30
Amazon Rekognition API 권한 참조	31
이미지 작업	35
이미지	35
이미지 방향 수정	36
이미지 작업의 모범 사례	36

Amazon Rekognition Image 작업 지연 시간	36
얼굴 인식 입력 이미지에 대한 권장 사항	36
Amazon S3 버킷 사용하기	37
로컬 파일 시스템 사용	40
JavaScript 사용	44
이미지 방향 및 경계 상자 좌표 가져오기	47
이미지의 방향 찾기	48
경계 상자 표시	49
예제: 이미지 방향 및 이미지의 경계 상자 좌표 가져오기	51
저장된 비디오 작업	57
감지 및 인식 유형	57
Amazon Rekognition Video API 개요	57
비디오 형식과 스토리지	58
사람 검색	58
Amazon Rekognition Video 작업 불러오기	59
비디오 분석 시작	59
Amazon Rekognition Video 분석 요청의 완료 상태 가져오기	60
Amazon Rekognition Video 분석 결과 가져오기	61
Amazon Rekognition Video 구성	64
여러 Amazon SNS 주제에 대한 액세스 권한 부여	65
기존 Amazon SNS 주제에 대한 액세스 권한 부여	66
비디오 분석(SDK)	66
사전 조건	67
비디오 분석(AWS CLI)	72
사전 조건	67
자습서: Amazon Rekognition Lambda 함수 생성	75
사전 조건	75
SNS 주제 생성	75
Lambda 함수 생성	76
Lambda 함수 구성	76
IAM Lambda 규칙 구성	76
AWS Toolkit for Eclipse Lambda 프로젝트 만들기	77
Lambda 함수 테스트	80
참조: 비디오 분석 결과 알림	81
Amazon Rekognition Video 문제 해결	82
Amazon SNS 주제로 전송된 완료 상태를 수신할 수 없습니다.	82
스트리밍 비디오 작업	84
스트리밍 비디오에서 얼굴 인식	84
사전 조건	85
Kinesis 스트림 액세스 권한 부여	85
Kinesis Video Streams와 Kinesis Data Streams에 대한 액세스 허용	85
개별 Kinesis 스트림에 대한 액세스 허용	86
스트리밍 비디오 분석 시작	86
Amazon Rekognition Video 스트림 프로세서 만들기	88
Amazon Rekognition Video 스트림 프로세서 시작	89
스트림 프로세서 사용	89
Amazon Rekognition Video으로 비디오 스트리밍	92
분석 결과 읽기	93
Kinesis Video Stream을 Kinesis Data Stream으로 매핑	94
참조: Kinesis 얼굴 인식 레코드	95
JSON 레코드	96
InputInformation	97
KinesisVideo	97
StreamProcessorInformation	97
FaceSearchResponse	98
DetectedFace	98
MatchedFace	99

객체 및 장면 감지	100
이미지에서 레이블 감지	100
DetectLabels 작업 요청	101
DetectLabels 응답	101
비디오에서 레이블 감지	102
GetLabelDetection 작업 응답	102
얼굴 감지 및 분석	105
이미지에서 얼굴 감지	106
DetectFaces 작업 요청	109
DetectFaces 작업 응답	109
이미지에 있는 얼굴 비교	114
CompareFaces 작업 요청	119
CompareFaces 작업 응답	119
저장된 비디오에서 얼굴 감지	121
GetFaceDetection 작업 응답	124
모음에서 얼굴 검색	127
모음 관리	127
모음에서 얼굴 관리	127
모음에 있는 얼굴 검색	128
모음 만들기	128
CreateCollection 작업 요청	131
CreateCollection 작업 응답	131
모음 나열	131
ListCollections 작업 요청	133
ListCollections 작업 응답	134
모음 삭제	134
DeleteCollection 작업 요청	137
DeleteCollection 작업 응답	137
모음에 얼굴 추가	137
IndexFaces 작업 요청	141
IndexFaces 작업 응답	141
모음에 얼굴 나열	146
ListFaces 작업 요청	148
ListFaces 작업 응답	149
모음에서 얼굴 삭제	149
DeleteFaces 작업 요청	152
DeleteFaces 작업 응답	152
얼굴(얼굴 ID) 검색	152
SearchFaces 작업 요청	155
SearchFaces 작업 응답	155
얼굴(이미지) 검색	156
SearchFacesByImage 작업 요청	159
SearchFacesByImage 작업 응답	159
저장된 비디오에서 얼굴 검색	160
GetFaceSearch 작업 응답	163
인물 추적	167
GetPersonTracking 작업 응답	169
유명 인사 인식	173
이미지 속 유명 인사 인식	173
RecognizeCelebrities 호출	173
RecognizeCelebrities 작업 요청	177
RecognizeCelebrities 작업 응답	177
저장된 비디오 속 유명 인사 인식	179
GetCelebrityRecognition 작업 응답	181
유명 인사 정보 획득	183
GetCelebrityInfo 호출	183
GetCelebrityInfo 작업 요청	185

비안전 콘텐츠 감지	185
안전하지 않은 이미지 감지(API)	187
이미지에서 안전하지 않은 콘텐츠 감지(SDK)	188
DetectModerationLabels 작업 요청	191
DetectModerationLabels 작업 응답	191
안전하지 않은 저장된 비디오 감지	192
GetContentModeration 작업 응답	194
텍스트 감지	196
이미지에서 텍스트 감지	196
DetectText 작업 요청	199
DetectText 작업 응답	200
모니터링	207
Monitoring	207
Rekognition에 CloudWatch 측정치 사용	207
Rekognition 측정치에 액세스	208
경보 만들기	209
Rekognition의 CloudWatch 측정치	210
Rekognition의 CloudWatch 측정치	210
Rekognition의 CloudWatch 차원	211
AWS CloudTrail을 사용하여 Amazon Rekognition API 호출 로깅	211
CloudTrail의 Amazon Rekognition 정보	211
예: Amazon Rekognition 로그 파일 항목	212
API Reference	217
HTTP 헤더	217
Actions	217
CompareFaces	219
CreateCollection	224
CreateStreamProcessor	227
DeleteCollection	230
DeleteFaces	232
DeleteStreamProcessor	234
DescribeCollection	236
DescribeStreamProcessor	239
DetectFaces	243
DetectLabels	247
DetectModerationLabels	251
DetectText	254
GetCelebrityInfo	257
GetCelebrityRecognition	259
GetContentModeration	264
GetFaceDetection	268
GetFaceSearch	273
GetLabelDetection	278
GetPersonTracking	282
IndexFaces	287
ListCollections	295
ListFaces	298
ListStreamProcessors	301
RecognizeCelebrities	304
SearchFaces	308
SearchFacesByImage	311
StartCelebrityRecognition	315
StartContentModeration	318
StartFaceDetection	322
StartFaceSearch	326

StartLabelDetection	330
StartPersonTracking	334
StartStreamProcessor	337
StopStreamProcessor	339
Data Types	340
AgeRange	342
Beard	343
BoundingBox	344
Celebrity	346
CelebrityDetail	347
CelebrityRecognition	349
ComparedFace	350
ComparedSourceImageFace	351
CompareFacesMatch	352
ContentModerationDetection	353
Emotion	354
Eyeglasses	355
EyeOpen	356
Face	357
FaceDetail	359
FaceDetection	362
FaceMatch	363
FaceRecord	364
FaceSearchSettings	365
Gender	366
Geometry	367
Image	368
ImageQuality	369
Instance	370
KinesisDataStream	371
KinesisVideoStream	372
Label	373
LabelDetection	375
Landmark	376
ModerationLabel	377
MouthOpen	378
Mustache	379
NotificationChannel	380
Parent	381
PersonDetail	382
PersonDetection	383
PersonMatch	384
Point	385
Pose	386
S3Object	387
Smile	388
StreamProcessor	389
StreamProcessorInput	390
StreamProcessorOutput	391
StreamProcessorSettings	392
Sunglasses	393
TextDetection	394
UnindexedFace	396
Video	397
VideoMetadata	398
제한	400
Amazon Rekognition Image	400

Amazon Rekognition Video 저장된 비디오	400
Amazon Rekognition Video 스트리밍 비디오	401
문서 기록	402
AWS Glossary	404

Amazon Rekognition이 무엇입니까?

Amazon Rekognition은 이미지와 비디오 분석을 애플리케이션에 쉽게 추가할 수 있도록 해줍니다. Rekognition API에 이미지나 비디오를 제공하면 서비스에서 객체, 사람, 텍스트, 장면 및 활동을 파악할 수 있습니다. 부적절한 콘텐츠를 감지할 수도 있습니다. Amazon Rekognition도 매우 정확한 얼굴 분석과 얼굴 인식을 제공합니다. 사용자 확인, 카탈로그 작성, 인원 계산 및 공공 안전을 포함하여 다양한 사용 사례에서 얼굴을 탐지, 분석 및 비교할 수 있습니다.

Amazon Rekognition은 Amazon의 컴퓨터 비전 과학자들이 매일 수십억 개의 이미지와 비디오를 매일 분석 할 목적으로 개발하여 성능이 검증되었을 뿐만 아니라 확장성까지 뛰어난 딥 러닝 기술을 기반으로 하고 있습니다. 따라서 기계 학습 전문 지식이 필요하지 않습니다. Amazon Rekognition에는 Amazon S3에 저장된 모든 이미지 또는 비디오 파일을 신속하게 분석할 수 있는 간편하고 사용하기 쉬운 API가 포함되어 있습니다. Amazon Rekognition은 항상 새로운 데이터를 통해 학습하고 있으며, 저희도 지속적으로 새로운 라벨과 얼굴 인식 기능을 서비스에 추가하고 있습니다. 자세한 내용은 [Amazon Rekognition FAQ](#)를 참조하십시오.

Amazon Rekognition의 일반적인 사용 사례는 다음과 같습니다.

- 검색 가능한 이미지 및 비디오 라이브러리 – Amazon Rekognition은 이미지와 저장된 비디오를 검색할 수 있도록 만들어 이미지 내에 나타나는 객체와 장면을 발견할 수 있습니다.
- 얼굴 기반 사용자 확인 – 애플리케이션은 Amazon Rekognition을 통해 사용자 라이브 이미지와 참조 이미지를 비교하여 사용자 자격 증명을 확인할 수 있습니다.
- 감성 및 인구통계학 분석 – Amazon Rekognition은 얼굴 이미지에서 행복, 슬픔 또는 놀람 같은 감정과 성별 등 인구통계학 정보를 감지합니다. Rekognition은 스토어 내 위치 및 유사 시나리오와 같은 추세를 주기적으로 보고하기 위해 이미지를 분석하여 감성 및 인구통계학 속성을 Amazon Redshift에 전송합니다.
- 얼굴 인식 – Amazon Rekognition을 통해 얼굴 모음으로 알려진 컨테이너에 저장된 것과 일치하는 얼굴의 이미지, 저장된 비디오 및 스트리밍 비디오를 검색할 수 있습니다. 얼굴 모음은 소유하고 관리하는 얼굴 인덱스입니다. 얼굴을 기반으로 한 사용자 식별은 Amazon Rekognition에서 다음 두 가지 주요 단계가 필요 합니다.
 1. 얼굴 인덱싱.
 2. 얼굴 검색.
- 비안전 콘텐츠 감지 – Amazon Rekognition은 이미지에서 노골적이고 선정적인 성인 콘텐츠를 감지할 수 있습니다. 개발자는 반환된 메타데이터를 사용하여 비즈니스 요구를 기반으로 부적절한 콘텐츠를 필터링 할 수 있습니다. 이 API는 성인 콘텐츠의 존재를 기반으로 이미지에 플래그를 지정하는 것 외에 신뢰도 점수와 함께 계층적 레이블 목록도 반환합니다. 이러한 레이블은 성인 콘텐츠의 특정 범주를 나타내므로 대량의 사용자 생성 콘텐츠(UGC)의 세분화된 필터링 및 관리가 가능합니다. 예: 소설 및 데이트 사이트, 사진 공유 플랫폼, 블로그 및 포럼, 어린이용 앱, 전자 상거래 사이트, 엔터테인먼트 및 온라인 광고 서비스.
- 유명 인사 인식 – Amazon Rekognition은 이미지와 저장된 비디오에서 유명 인사를 인식할 수 있습니다. Rekognition은 엔터테인먼트와 미디어, 스포츠, 비즈니스, 엔터테인먼트, 미디어 등 다양한 범주에 걸쳐 수 천 명의 유명 인사 얼굴을 인식할 수 있습니다.

- 텍스트 감지 – Amazon Rekognition Text in Image를 이용해 이미지에서 텍스트 콘텐츠를 인식하고 추출할 수 있습니다. Text in Image는 고도로 정형화된 이미지를 포함한 대부분의 글꼴을 지원합니다. 배너 및 포스터에서 흔히 볼 수 있는 것과 같이 방향이 다른 텍스트와 숫자를 감지합니다. 이미지 공유와 소셜 미디어 애플리케이션에서 같은 키워드가 포함된 이미지 인덱스를 바탕으로 시각적 검색을 활성화할 수 있습니다. 미디어 및 엔터테인먼트 애플리케이션에서는 광고, 뉴스, 스포츠 드레스, 자막처럼 화면 위 관련된 텍스트를 기반으로 영상 목록을 작성할 수 있습니다. 마지막으로, 공공 안전 애플리케이션에서 길거리 카메라로 찍은 이미지에서 자동차 번호판을 바탕으로 차량을 식별할 수 있습니다.

Amazon Rekognition을 사용하면 다음과 같은 이점이 있습니다.

- 강력한 이미지 및 비디오 인식을 앱에 통합 – Amazon Rekognition은 간단한 API로 강력하고 정확한 이미지 분석을 사용할 수 있으므로 복잡하게 애플리케이션에 이미지 인식 기능을 포함 시킬 필요가 없습니다. Rekognition의 안정적인 이미지 및 비디오 분석을 활용하는 데는 컴퓨터 비전이나 딥 러닝 전문성이 필요 없습니다. Rekognition의 API를 사용하면 어떤 웹, 모바일 또는 연결된 장치의 애플리케이션에도 쉽고 빠르게 이미지 및 비디오 분석을 통합할 수 있습니다.
- 딥 러닝 기반 이미지 및 비디오 분석 – Rekognition은 딥 러닝 기술을 사용하여 이미지를 정확하게 분석하고, 이미지의 얼굴을 찾아 비교하며, 이미지와 비디오 내에서 객체와 장면을 감지합니다.
- 확장 가능한 이미지 분석 – Amazon Rekognition을 통해 수백만 개의 이미지를 분석할 수 있으므로 엄청난 양의 시각적 데이터를 큐레이팅하고 정리할 수 있습니다.
- 다른 AWS 서비스와의 통합 – Amazon Rekognition은 Amazon S3 및 AWS Lambda 같은 다른 AWS 서비스와 완벽하게 연동되도록 만들어졌습니다. Amazon S3 이벤트에 응답하여 Rekognition의 API를 Lambda에서 직접 호출할 수 있습니다. Amazon S3와 Lambda는 애플리케이션의 수에 반응해 자동으로 조정되므로 확장 가능하고 저렴하며 안정적인 이미지 분석 애플리케이션을 구축할 수 있습니다. 예를 들어 집에 사람이 도착할 때마다 도어 카메라가 방문객의 사진을 Amazon S3에 업로드하여 Rekognition API 작업을 사용해 손님을 식별하는 Lambda 기능을 트리거합니다. 데이터를 로드하거나 이동할 필요 없이 Amazon S3에 저장된 이미지에서 직접 분석을 실행할 수 있습니다. AWS Identity and Access Management(IAM) 지원으로 Rekognition API 작업에 대한 액세스를 안전하게 제어하기 쉽습니다. IAM을 사용하면 AWS 사용자 및 그룹을 만들고 관리하여 적절한 액세스 권한을 개발자와 최종 사용자에게 부여할 수 있습니다.
- 저렴한 비용 – Amazon Rekognition을 사용하면 분석하는 이미지와 비디오 수와 저장하는 얼굴 메타데이터에 대한 비용만 지불하면 됩니다. 최소 요금이나 사전 약정은 없으며, 무료로 사용을 시작한 다음 확장에 따라 Rekognition의 계층화된 요금 모델을 활용하여 더 많은 비용을 절감하십시오.

Amazon Rekognition 및 HIPAA 자격

이것은 HIPAA 적격 서비스입니다. AWS, 미국 HIPAA(Health Insurance Portability and Accountability Act of 1996), AWS 서비스를 이용한 보호 대상 건강 정보(PHI)의 처리, 저장, 전송에 대한 자세한 정보는 [HIPAA 개요](#)를 확인하십시오.

Amazon Rekognition을 처음 사용하십니까?

Amazon Rekognition을 처음 사용하는 경우, 먼저 다음 단원을 순서대로 읽어보십시오.

1. [Amazon Rekognition: 작동 방식 \(p. 4\)](#) - 이 단원에서는 종단 간 경험 생성에 사용하는 다양한 Amazon Rekognition 구성 요소를 소개합니다.

2. [Amazon Rekognition 시작하기 \(p. 10\)](#) - 이 단원에서는 계정을 설정하고 Amazon Rekognition API를 테스트합니다.
3. [이미지 작업 \(p. 35\)](#)이 단원에서는 Amazon S3 버킷에 저장된 이미지와 로컬 파일 시스템에서 로드된 이미지에서 Amazon Rekognition을 사용하는 방법에 대한 정보를 다룹니다.
4. [저장된 비디오 작업 \(p. 57\)](#)이 단원에서는 Amazon S3 버킷에 저장된 비디오에서 Amazon Rekognition를 사용하는 방법에 대한 정보를 다룹니다.
5. [스트리밍 비디오 작업 \(p. 84\)](#)이 단원에서는 스트리밍 비디오에서 Amazon Rekognition를 사용하는 방법에 대한 정보를 다룹니다.

Amazon Rekognition: 작동 방식

Amazon Rekognition은 API 세트 2개를 제공합니다. 즉, 이미지를 분석하는 Amazon Rekognition Image와 비디오를 분석하는 Amazon Rekognition Video입니다.

API는 이미지와 비디오의 감지과 인식 분석을 수행하여 애플리케이션에서 사용할 수 있는 분석 정보를 제공합니다. 예를 들어 Amazon Rekognition Image를 사용하여 사진 관리 애플리케이션의 고객 경험을 향상시킬 수 있습니다. 고객이 사진을 업로드하면 애플리케이션에서 Amazon Rekognition Image를 사용하여 이미지의 실제 객체 또는 얼굴을 감지합니다. 애플리케이션이 Amazon Rekognition Image에서 반환한 정보를 저장하면 사용자는 특정 객체 또는 얼굴이 있는 사진을 자신의 사진 모음에 퀴리할 수 있습니다. 보다 심화된 퀴리도 가능합니다. 예를 들어 사용자는 웃고 있는 얼굴에 대해 퀴리하거나 특정 연령의 얼굴을 퀴리할 수 있습니다.

Amazon Rekognition Video는 저장된 비디오 전체에서 특정 사람이 감지된 부분을 추적할 수 있습니다. 또는 Amazon Rekognition Video를 사용하여 스트리밍 비디오에서 얼굴 설명이 Amazon Rekognition에 이미 저장된 얼굴 설명과 일치하는 사람을 검색할 수 있습니다.

Amazon Rekognition API를 통해 딥 러닝 이미지 분석 사용이 쉽습니다. 예를 들어, [RecognizeCelebrities \(p. 304\)](#)는 이미지에서 감지된 최대 100명의 정보를 반환합니다. 여기에는 이미지에서 유명 인사 얼굴이 감지된 부분에 대한 정보와 유명 인사에 대한 추가 정보를 가져올 부분이 포함됩니다.

다음 정보에는 Amazon Rekognition에서 제공하는 분석 유형과 Amazon Rekognition Image 및 Amazon Rekognition Video 작업에 대한 개요가 포함됩니다. 또한 비스토리지와 스토리지 작업의 차이도 없어집니다.

항목

- [감지 및 인식 유형 \(p. 4\)](#)
- [이미지 및 비디오 작업 \(p. 5\)](#)
- [비스토리지 및 스토리지 API 작업 \(p. 7\)](#)
- [모델 버전 관리 \(p. 9\)](#)

감지 및 인식 유형

다음은 Amazon Rekognition Image API와 Amazon Rekognition Video API가 수행할 수 있는 감지과 인식 유형입니다. API에 대한 정보는 [이미지 및 비디오 작업 \(p. 5\)](#) 단원을 참조하십시오.

레이블

레이블은 객체(예: 꽃, 나무 또는 테이블), 이벤트(예: 결혼식, 졸업식 또는 생일 파티), 개념(예: 풍경, 저녁, 자연), 활동(예: 차에서 내리기) 중 어느 것이든 지정할 수 있습니다. Amazon Rekognition은 이미지와 비디오에서 레이블을 감지할 수 있습니다. 그렇지만 활동은 이미지에서 감지되지 않습니다. 자세한 내용은 [객체 및 장면 감지 \(p. 100\)](#) 단원을 참조하십시오.

이미지에서 레이블을 감지하려면 [DetectLabels \(p. 247\)](#)를 사용합니다. 저장된 비디오에서 레이블을 감지하려면 [StartLabelDetection \(p. 330\)](#)을 사용합니다.

얼굴

Amazon Rekognition은 이미지와 저장된 비디오에서 얼굴을 감지할 수 있습니다. Amazon Rekognition을 통해 이미지에서 얼굴을 감지한 위치에 대한 정보뿐 아니라 눈의 위치와 같은 얼굴 표식과 행복함 또는 슬픔과 같은 감지된 감정에 대한 정보를 가져올 수 있습니다. 원본 이미지의 한 얼굴과 다른 이미지에서 감지된 얼굴

도 비교할 수 있습니다. 얼굴에 대한 정보도 저장하여 다중에 검색할 수 있습니다. 자세한 내용은 [얼굴 감지 및 분석 \(p. 105\)](#) 단원을 참조하십시오.

이미지에서 얼굴을 감지하려면 [DetectFaces \(p. 243\)](#)를 사용합니다. 비디오에 저장된 얼굴을 감지하려면 [StartFaceDetection \(p. 322\)](#)를 사용합니다.

얼굴 검색

Amazon Rekognition은 얼굴 검색할 수 있습니다. 얼굴 정보는 모음이라고 하는 컨테이너로 인덱싱됩니다. 그런 다음 모음의 얼굴 정보는 이미지, 저장된 비디오, 스트리밍 비디오에서 감지된 얼굴과 일치시킬 수 있습니다. 자세한 내용은 [모음에서 얼굴 검색 \(p. 127\)](#) 단원을 참조하십시오.

이미지에서 알려진 얼굴을 감지하려면 [DetectFaces \(p. 243\)](#)를 사용합니다. 저장된 비디오에서 알려진 얼굴을 감지하려면 [StartFaceDetection \(p. 322\)](#)를 사용합니다. 스트리밍 비디오에서 알려진 얼굴을 검색하려면 [CreateStreamProcessor \(p. 227\)](#)를 사용합니다.

사람

Amazon Rekognition은 저장된 비디오에서 사람을 추적할 수 있습니다. Amazon Rekognition Video는 비디오에서 감지된 사람에 대한 추적, 얼굴 세부 정보, 프레임 위치 정보를 제공합니다. 이미지에서 사람을 감지할 수 없습니다. 자세한 내용은 [인물 추적 \(p. 167\)](#) 단원을 참조하십시오.

저장된 비디오에서 사람을 감지하려면 [StartPersonTracking \(p. 334\)](#)를 사용합니다.

유명 인사

Amazon Rekognition은 이미지와 저장된 비디오에서 수천 명의 유명 인사를 인식할 수 있습니다. 이미지에서 유명 인사 얼굴이 감지된 부분, 얼굴 표식 및 유명 인사의 얼굴 표정에 대한 정보를 가져올 수 있습니다. 저장된 비디오 전체에 나타나는 유명 인사에 대한 추적 정보를 가져올 수 있습니다. 또한 인식된 유명 인사에 대한 추가 정보를 가져올 수도 있습니다. 자세한 내용은 [유명 인사 인식 \(p. 173\)](#) 단원을 참조하십시오.

이미지에서 유명 인사를 인식하려면 [RecognizeCelebrities \(p. 304\)](#)를 사용합니다. 저장된 비디오에서 유명 인사를 인식하려면 [StartCelebrityRecognition \(p. 315\)](#)를 사용합니다.

텍스트 감지

Amazon Rekognition Text in Image는 이미지의 텍스트를 감지하여 머신 판독 가능한 텍스트로 변환합니다. 자세한 내용은 [텍스트 감지 \(p. 196\)](#) 단원을 참조하십시오.

이미지에서 텍스트를 감지하려면 [DetectText \(p. 254\)](#)를 사용합니다.

안전하지 않은 콘텐츠

Amazon Rekognition은 이미지와 저장된 비디오에서 노골적이거나 선정적인 성인 콘텐츠를 분석할 수 있습니다. 자세한 내용은 [비안전 콘텐츠 감지 \(p. 187\)](#) 단원을 참조하십시오.

안전하지 않은 이미지를 감지하려면 [DetectModerationLabels \(p. 251\)](#)를 사용합니다. 안전하지 않은 저장된 비디오를 감지하려면 [StartContentModeration \(p. 318\)](#)을 사용합니다.

이미지 및 비디오 작업

Amazon Rekognition은 API 세트 2개를 제공합니다. 즉, 이미지를 분석하는 Amazon Rekognition Image와 저장된 비디오와 스트리밍 비디오를 분석하는 Amazon Rekognition Video입니다. 다음 주제에서 각 API 세트를 간략하게 설명합니다.

Amazon Rekognition Image 및 Amazon Rekognition Video API는 얼굴이나 객체와 같은 다양한 개체를 감지하거나 인식할 수 있습니다. 지원되는 인식 및 감지 유형에 대한 정보는 [감지 및 인식 유형 \(p. 4\)](#) 단원을 참조하십시오.

Amazon Rekognition Image 작업

Amazon Rekognition 이미지 작업은 동기식으로 수행됩니다. 입력과 응답은 JSON 형식입니다. Amazon Rekognition Image 작업은 .jpg 또는 .png 이미지 형식의 입력 이미지를 분석합니다. Amazon Rekognition Image 작업에 전달된 이미지는 Amazon S3 버킷에 저장할 수 있습니다. AWS CLI를 사용하지 않는 경우 byte64 인코딩 이미지 바이트를 Amazon Rekognition 작업에 직접 전달할 수도 있습니다. 자세한 내용은 [이미지 작업 \(p. 35\)](#) 단원을 참조하십시오.

Amazon Rekognition Video 작업

Amazon Rekognition Video는 Amazon S3 버킷에 저장된 비디오와 Amazon Kinesis Video Streams을 통해 스트리밍된 비디오를 분석할 수 있습니다.

Amazon Rekognition Video 비디오 작업은 비동기식으로 수행됩니다. Amazon Rekognition Video 스토리지 비디오 작업을 사용하면 원하는 분석 유형에 대한 시작 작업을 호출하여 분석을 시작할 수 있습니다. 예를 들어, 저장된 비디오에서 얼굴을 감지하려면 [StartFaceDetection \(p. 322\)](#)을 호출합니다. 완료되면 Amazon Rekognition은 Amazon SNS 주제에 완료 상태를 게시합니다. 분석 작업 결과를 가져오려면 요청한 분석 유형에 대한 가져오기 작업을 호출합니다. 예 [GetFaceDetection \(p. 268\)](#). 자세한 내용은 [저장된 비디오 작업 \(p. 57\)](#) 단원을 참조하십시오.

Amazon Rekognition Video 스트리밍 비디오 작업을 통해 Amazon Rekognition Video 모음에 저장된 얼굴을 검색할 수 있습니다. Amazon Rekognition Video는 Kinesis video stream을 분석하고 검색 결과를 Kinesis data stream에 출력합니다. Amazon Rekognition Video 스트림 프로세서를 만들고 사용하여 비디오 분석을 관리합니다. 예를 들어 [CreateStreamProcessor \(p. 227\)](#)를 호출하여 스트림 프로세서를 만들 수 있습니다. 자세한 내용은 [스트리밍 비디오 작업 \(p. 84\)](#) 단원을 참조하십시오.

비스토리지 및 스토리지 기반 작업

Amazon Rekognition 작업은 다음과 같은 범주로 그룹화됩니다.

- **비 스토리지 API 작업** – 이 작업에서 Amazon Rekognition은 어떤 정보도 유지하지 않습니다. 입력 이미지와 비디오를 제공하고, 이 작업은 분석을 실시하고 결과를 반환하지만 Amazon Rekognition에는 아무것도 저장되지 않습니다. 자세한 내용은 [비스토리지 작업 \(p. 7\)](#) 단원을 참조하십시오.
- **스토리지 기반 API 작업** – Amazon Rekognition 서버는 감지된 얼굴 정보를 모음으로 알려진 컨테이너에 저장할 수 있습니다. Amazon Rekognition은 유지되는 얼굴 정보에서 얼굴 일치를 검색하는데 사용할 수 있는 추가 API 작업을 제공합니다. 자세한 내용은 [스토리지 기반 API 작업 \(p. 8\)](#) 단원을 참조하십시오.

AWS SDK 또는 HTTP를 사용하여 Amazon Rekognition API 작업 호출

AWS SDK를 사용하거나 HTTP를 사용하여 직접 Amazon Rekognition API 작업을 호출할 수 있습니다. 특별한 이유가 없다면 항상 AWS SDK를 사용해야 합니다. 이 단원의 Java 예제는 [AWS SDK](#)를 사용합니다. Java 프로젝트 파일은 제공되지 않지만 [AWS Toolkit for Eclipse](#)를 사용하여 Java를 사용하는 AWS 애플리케이션을 개발할 수 있습니다.

이 단원의 .NET 예제는 [.NET용 AWS SDK](#)를 사용합니다. [AWS Toolkit for Visual Studio](#)를 사용하면 .NET을 사용하는 AWS 애플리케이션을 개발할 수 있습니다. 여기에는 애플리케이션 배포 및 서비스 관리를 위한 AWS Explorer와 유용한 템플릿이 포함됩니다. AWS의 .NET 개발자의 경우 [.NET 개발자를 위한 AWS 가이드](#)를 참조하십시오.

이 가이드의 [API Reference \(p. 217\)](#)에서는 HTTP를 사용한 Amazon Rekognition 작업 호출을 다룹니다.
Java 참조 정보는 [AWS SDK for Java](#)을 참조하십시오.

사용 가능한 Amazon Rekognition 서비스 앤드포인트는 [AWS 리전 및 앤드포인트](#)에 나와 있습니다.

HTTP를 사용하여 Amazon Rekognition을 호출할 때는 POST HTTP 작업을 사용합니다.

비스토리지 및 스토리지 API 작업

Amazon Rekognition는 두 가지 유형의 API 작업을 제공합니다. 즉, Amazon Rekognition에 저장된 정보가 없는 비스토리지 작업과 특정 얼굴 정보가 Amazon Rekognition에 저장된 스토리지 작업입니다.

비스토리지 작업

Amazon Rekognition은 이미지를 대상으로 다음과 같은 비스토리지 API 작업을 제공합니다.

- [DetectLabels \(p. 247\)](#)
- [DetectFaces \(p. 243\)](#)
- [CompareFaces \(p. 219\)](#)
- [DetectModerationLabels \(p. 251\)](#)
- [RecognizeCelebrities \(p. 304\)](#)
- [DetectText \(p. 254\)](#)

Amazon Rekognition은 비디오를 대상으로 다음과 같은 비스토리지 API 작업을 제공합니다.

- [StartLabelDetection \(p. 330\)](#)
- [StartFaceDetection \(p. 322\)](#)
- [StartPersonTracking \(p. 334\)](#)
- [StartCelebrityRecognition \(p. 315\)](#)
- [StartContentModeration \(p. 318\)](#)

이러한 작업을 비 스토리지 API 작업이라고 하는 이유는 작업을 호출할 때 Amazon Rekognition이 입력 이미지에 관해 발견된 어떤 정보도 유지하지 않기 때문입니다. 다른 모든 Amazon Rekognition API 작업과 마찬가지로 비 스토리지 API 작업은 입력 이미지 바이트를 유지하지 않습니다.

다음 예제 시나리오는 애플리케이션에 비 스토리지 API 작업을 통합할 수 있는 경우를 보여 줍니다. 이 시나리오에서는 로컬 이미지 리포지토리가 있다고 가정합니다.

Example 1: 로컬 리포지토리에서 특정 레이블이 포함된 이미지를 찾는 애플리케이션

먼저 리포지토리에 있는 각 이미지에서 Amazon Rekognition `DetectLabels` 작업을 사용하여 레이블을 감지하고 다음과 같이 클라이언트 측 인덱스를 빌드합니다.

Label	ImageID
tree	image-1
flower	image-1
mountain	image-1
tulip	image-2
flower	image-2
apple	image-3

그러면 애플리케이션이 이 인덱스를 검색해 로컬 리포지토리에서 특정 레이블이 포함된 이미지를 찾을 수 있습니다. 예를 들어 나무가 포함된 이미지를 표시하십시오.

Amazon Rekognition이 감지하는 각 레이블에는 연결된 신뢰도 값이 있습니다. 신뢰도 값은 입력 이미지에 해당 레이블이 포함될 신뢰도 수준을 나타냅니다. 이 신뢰도 값을 사용하여 필요하다면 감지의 신뢰도 수준에 대한 애플리케이션 요구 사항에 따라 레이블에서 추가적인 클라이언트 측 필터링을 수행할 수 있습니다. 예를 들어 정확한 레이블이 필요하다면 신뢰도가 더 높은(95% 이상) 레이블만을 필터링하고 선택할 수 있습니다. 애플리케이션에 이보다 높은 신뢰도 값이 필요하지 않은 경우에는 신뢰도 값이 보다 낮은(50% 정도) 레이블을 필터링할 수 있습니다.

Example 2: 향상된 얼굴 이미지를 표시하는 애플리케이션

먼저 Amazon Rekognition DetectFaces 작업을 사용하여 로컬 리포지토리에 있는 각 이미지에서 얼굴을 탐지하고 클라이언트 측 인덱스를 빌드할 수 있습니다. 이 작업은 각 얼굴에 대해 경계 상자, 얼굴 표식(예: 입과 귀의 위치), 얼굴 속성(예: 성별)이 포함된 메타데이터를 반환합니다. 이 메타데이터는 다음과 같이 클라이언트 측 로컬 인덱스에 저장할 수 있습니다.

ImageID	FaceID	FaceMetaData
image-1	face-1	<boundingbox>, etc.
image-1	face-2	<boundingbox>, etc.
image-1	face-3	<boundingbox>, etc.
...		

이 인덱스에서 기본 키는 ImageID와 FaceID의 조합입니다.

그런 다음 애플리케이션이 로컬 리포지토리에서 이미지를 표시할 때 인덱스의 정보를 사용하여 이미지를 향상시킬 수 있습니다. 예를 들어 얼굴 주위에 경계 상자를 추가하거나 얼굴 특징을 강조 표시할 수 있습니다.

스토리지 기반 API 작업

Amazon Rekognition Image는 이미지에서 얼굴을 검색하고 Amazon Rekognition 모음에서 감지된 얼굴 특징에 대한 정보를 유지하는 데 사용할 수 있는 [IndexFaces \(p. 287\)](#) 작업을 지원합니다. 이것이 스토리지 기반 API 작업의 예인 이유는 서비스가 서버에서 정보를 유지하기 때문입니다.

Amazon Rekognition Image는 다음과 같은 비 스토리지 API 작업을 제공합니다.

- [IndexFaces \(p. 287\)](#)
- [ListFaces \(p. 298\)](#)
- [SearchFacesByImage \(p. 311\)](#)
- [SearchFaces \(p. 308\)](#)
- [DeleteFaces \(p. 232\)](#)

Amazon Rekognition Video는 다음과 같은 스토리지 API 작업을 제공합니다.

- [StartFaceSearch \(p. 326\)](#)
- [CreateStreamProcessor \(p. 227\)](#)

얼굴 정보를 저장하려면 먼저 계정의 AWS 리전 중 하나에서 얼굴 모음을 만들어야 합니다. [IndexFaces](#) 작업을 호출할 때 이 얼굴 모음을 지정합니다. 얼굴 모음을 만들고 모든 얼굴의 얼굴 특징 정보를 저장한 후에 모음에서 얼굴 일치를 검색할 수 있습니다. 예를 들어 이미지에서 가장 큰 얼굴을 감지하고 모음에서 일치하는 얼굴을 감지하려면 [searchFacesByImage](#).를 호출합니다.

[IndexFaces](#)를 통해 모음에 저장된 얼굴 정보는 Amazon Rekognition Video 작업에 액세스할 수 있습니다. 예를 들어 이미지에서 가장 큰 얼굴을 검색하고 모음에서 일치하는 얼굴을 검색하려면 [StartFaceSearch \(p. 326\)](#)를 호출합니다.

모음을 만들고 관리하는 방법에 대한 정보는 [모음에서 얼굴 검색 \(p. 127\)](#)를 참조하십시오.

Note

이 서비스는 실제 이미지 바이트를 유지하지 않습니다. 그 대신 기본 감지 알고리즘이 먼저 입력 이미지에서 얼굴을 감지하고 얼굴 특징을 각 얼굴의 특징 벡터로 추출한 다음 데이터베이스에 저장합니다. Amazon Rekognition은 얼굴 일치를 수행할 때 이러한 특징 벡터를 사용합니다.

Example 1: 건물 액세스를 인증하는 애플리케이션

먼저 얼굴을 추출해 검색 가능한 이미지 벡터로 저장하는 [IndexFaces](#) 작업을 사용하여 얼굴 모음을 만들어 스캔한 배지 이미지를 저장할 수 있습니다. 그런 다음 직원이 건물에 들어오면 직원 얼굴 이미지가 캡처되어 [SearchFacesByImage](#) 작업으로 전송됩니다. 얼굴 일치에서 충분히 높은 유사성 점수(가령 99%)가 나오면 직원을 인증할 수 있습니다.

모델 버전 관리

Amazon Rekognition은 딥 러닝 모델을 사용하여 얼굴 감지를 수행하고 모음에서 얼굴을 감지합니다. 그리고 계속해서 고객 피드백과 딥 러닝 연구 발전을 토대로 모델 정확도를 개선하고 있습니다. 이러한 개선 사항은 모델 업데이트로 전달됩니다. 예를 들어 모델 버전 1.0에서는 [IndexFaces \(p. 287\)](#)가 이미지에서 가장 큰 얼굴 15개를 인덱싱할 수 있습니다. 최신 버전의 모델을 통해 [IndexFaces](#)는 이미지에서 가장 큰 얼굴 100 개를 인덱싱할 수 있습니다.

새 모음을 만들 때 최신 버전의 모델과 연결됩니다. 정확도를 개선하기 위해 모델은 정기적으로 업데이트됩니다.

새 버전의 모델이 릴리스되면 다음 상황이 나타납니다.

- 만든 새 모음은 최신 모델과 연결됩니다. [IndexFaces \(p. 287\)](#)를 사용하여 새 모음에 추가하는 얼굴은 최신 모델을 사용하여 감지됩니다.
- 기존 모음은 모음을 만들 때 사용했던 모델 버전을 계속 사용합니다. 이 모음에 저장된 얼굴 벡터는 최신 모델 버전으로 자동 업데이트되지 않습니다.
- 기존 모음에 추가되는 새 얼굴은 이미 모음에 연결된 모델을 사용하여 감지합니다.

다른 모델 버전과는 호환되지 않습니다. 특히 서로 다른 모델 버전을 사용하는 여러 모음으로 이미지를 인덱싱하는 경우 감지된 동일한 얼굴의 얼굴 식별자는 서로 다릅니다. 동일한 모델과 연결된 여러 모음으로 이미지가 인덱싱되는 경우 얼굴 식별자는 동일합니다.

모음 관리에서 모델 업데이트를 고려하지 않을 경우 애플리케이션이 호환성 문제를 겪을 수 있습니다. 모음 작업 응답을 통해 반환된 `FaceModelVersion` 필드를 사용하여 모음이 사용하는 모델의 버전을 확인할 수 있습니다. 예: `CreateCollection`.

모음에 있는 기존 얼굴 벡터는 최신 모델 버전으로 자동 업데이트되지 않습니다. Amazon Rekognition은 소스 이미지 바이트를 저장하지 않기 때문에 최신 버전의 모델을 사용하여 이미지를 자동으로 다시 인덱싱할 수 없습니다.

기존 모음에 저장된 얼굴에서 최신 모델을 사용하려면 새 모음([CreateCollection \(p. 224\)](#))을 만들고 소스 이미지를 새 모음([IndexFaces](#))으로 다시 인덱싱합니다. 새 모음의 얼굴 식별자는 이전 모음의 얼굴 식별자와 다르기 때문에 사용 중인 애플리케이션에서 저장한 모든 얼굴 식별자를 업데이트해야 합니다. 이전 모음이 더 이상 필요하지 않으면 [DeleteCollection \(p. 230\)](#)을 사용하여 삭제할 수 있습니다.

Amazon Rekognition 시작하기

이 단원에서는 Amazon Rekognition 사용 시작하기에 대한 주제를 다룹니다. Amazon Rekognition을 처음 사용하는 경우, 먼저 [Amazon Rekognition: 작동 방식 \(p. 4\)](#)에 나와 있는 개념과 용어를 검토하는 것이 좋습니다.

항목

- [1단계: AWS 계정 설정 및 IAM 사용자 만들기 \(p. 10\)](#)
- [2단계: AWS CLI 및 AWS SDK 설정 \(p. 11\)](#)
- [3단계: AWS CLI 및 AWS SDK API를 사용하여 시작하기 \(p. 13\)](#)
- [4단계: Amazon Rekognition 콘솔 사용 시작하기 \(p. 13\)](#)

1단계: AWS 계정 설정 및 IAM 사용자 만들기

Amazon Rekognition을 처음 사용하기 전에 다음 작업을 완료해야 합니다.

1. [AWS에 가입 \(p. 10\)](#)
2. [IAM 사용자 생성 \(p. 11\)](#)

AWS에 가입

Amazon Web Services(AWS)에 가입하면 Amazon Rekognition을 포함해 AWS의 모든 서비스에 AWS 계정이 자동으로 등록됩니다. 사용한 서비스에 대해서만 청구됩니다.

Amazon Rekognition에서는 사용한 리소스에 대해서만 비용을 지불합니다. AWS를 처음 사용하는 고객인 경우, 무료로 Amazon Rekognition을 시작할 수 있습니다. 자세한 내용은 [AWS 프리 티어](#)를 참조하십시오.

이미 AWS 계정이 있다면 다음 작업으로 건너뛰십시오. AWS 계정이 없는 경우에는 다음 절차의 단계를 수행하여 계정을 만듭니다.

AWS 계정을 만들려면 다음을 수행합니다.

1. <https://aws.amazon.com/>을 열고 Create an AWS Account(AWS 계정 생성)를 선택합니다.

Note

전에 AWS 계정 루트 사용자 자격 증명을 사용하여 AWS Management 콘솔에 로그인한 적이 있는 경우 Sign in to a different account(다른 계정으로 로그인)를 선택합니다. 전에 IAM 자격 증명을 사용하여 콘솔에 로그인한 적이 있는 경우 Sign-in using root account credentials(루트 계정 자격 증명으로 로그인)를 선택합니다. 그런 다음 Create a new AWS account(새 AWS 계정 생성)를 선택합니다.

2. 온라인 지시 사항을 따릅니다.

등록 절차 중 전화를 받고 전화 키패드를 사용하여 확인 코드를 입력하는 과정이 있습니다.

다음 작업에 필요하므로 AWS 계정 ID를 기록합니다.

IAM 사용자 생성

Amazon Rekognition과 같은 AWS 서비스를 사용하려면 액세스할 때 자격 증명을 제공해야 합니다. 이를 통해 서비스가 소유한 리소스에 액세스할 수 있는 권한이 있는지를 확인합니다. 콘솔은 암호를 요구합니다. AWS 계정에 대한 액세스 키를 생성하면 AWS CLI 또는 API에 액세스할 수 있습니다. 그러나 AWS 계정용 자격 증명을 사용하여 AWS에 액세스하는 것은 권장되지 않습니다. 그 대신 다음과 같이 하는 것이 좋습니다.

- AWS Identity and Access Management(IAM)을 사용하여 IAM 사용자를 만듭니다.
- 관리 권한을 가진 IAM 그룹에 사용자를 추가합니다.

그러면 특정 URL이나 그 IAM 사용자의 자격 증명을 사용하여 AWS에 액세스할 수 있습니다.

AWS에 가입했지만 IAM 사용자를 만들지 않은 경우, IAM 콘솔을 사용하여 사용자를 만들 수 있습니다. 절차에 따라 계정에 IAM 사용자를 만듭니다.

IAM 사용자를 만들고 콘솔에 로그인하려면

1. AWS 계정에서 관리자 권한을 가진 IAM 사용자를 만듭니다. 관련 지침은 IAM 사용 설명서의 [IAM 사용자와 관리자 그룹 처음 만들기](#)를 참조하십시오.
2. IAM 사용자로 특정 URL을 사용하여 AWS Management 콘솔에 로그인합니다. 자세한 내용은 IAM 사용 설명서의 [사용자의 계정 로그인 방법](#)을 참조하십시오.

Note

관리자 권한을 가진 IAM 사용자는 해당 계정의 AWS 서비스에 제한 없이 액세스할 수 있습니다. Amazon Rekognition 작업에 대한 액세스를 제한하는 방법은 [Amazon Rekognition에 대한 자격 증명 기반 정책\(IAM 정책\) 사용 \(p. 28\)](#) 단원을 참조하십시오. 이 가이드의 코드 예제는 `AmazonRekognitionFullAccess` 권한을 가진 사용자가 있다는 전제하에서 진행됩니다. 예를 들어 Amazon S3 버킷에 저장된 이미지나 비디오를 액세스하려면 `AmazonS3ReadOnlyAccess`가 필요합니다. Amazon Rekognition Video 저장된 비디오 코드 예제는 또한 `AmazonSQSFullAccess` 권한도 필요로 합니다. 보안 요구 사항에 따라 이러한 권한으로 제한한 IAM 그룹을 사용할 수 있습니다. 자세한 내용은 [IAM 그룹 만들기](#)를 참조하십시오.

IAM에 대한 자세한 내용은 다음을 참조하십시오.

- [AWS Identity and Access Management\(IAM\)](#)
- [시작하기](#)
- [IAM 사용 설명서](#)

다음 단계

[2단계: AWS CLI 및 AWS SDK 설정 \(p. 11\)](#)

2단계: AWS CLI 및 AWS SDK 설정

다음 단계는 이 설명서의 예제에서 사용하는 AWS Command Line Interface(AWS CLI)와 AWS SDK를 설치하는 방법을 보여줍니다. AWS SDK 호출을 인증하는 방법에는 여러 가지가 있습니다. 이 가이드의 예제에서는 AWS CLI 명령과 AWS SDK API 작업을 호출하기 위해 기본 자격 증명 프로파일을 사용합니다.

사용할 수 있는 AWS 리전 목록은 [Amazon Web Services 일반 참조](#)의 리전 및 엔드포인트를 참조하십시오.

단계에 따라 AWS SDK를 다운로드하고 구성합니다.

AWS CLI와 AWS SDK를 설정하려면

1. 사용하려는 AWS CLI와 AWS SDK를 다운로드하여 설치합니다. 이 가이드에서는 AWS CLI, Java, Python, PHP, .NET, JavaScript용 예제를 제공합니다. 그 밖의 AWS SDK에 대한 자세한 정보는 [Amazon Web Services용 도구](#)를 참조하십시오.
 - [AWS CLI](#)
 - [AWS SDK for Java](#)
 - [Python용 AWS SDK\(Boto3\)](#)
 - [PHP용 AWS SDK](#)
 - [.NET용 AWS SDK](#)
 - [AWS SDK for JavaScript](#)
2. [IAM 사용자 생성](#) (p. 11)에서 만든 사용자의 액세스 키를 만듭니다.
 - a. AWS Management 콘솔에 로그인한 다음 <https://console.aws.amazon.com/iam/>에서 IAM 콘솔을 엽니다.
 - b. 탐색 창에서 [Users]를 선택합니다.
 - c. [IAM 사용자 생성](#) (p. 11)에서 만든 사용자의 이름을 선택합니다.
 - d. 액세스 키 생성을 선택합니다. 그런 다음 .csv 파일 다운로드를 선택하여 액세스 키 ID 및 보안 액세스 키를 컴퓨터에 CSV 파일로 저장합니다. 안전한 위치에 파일을 저장합니다. 이 대화 상자를 닫은 후에는 보안 액세스 키에 다시 액세스할 수 없습니다. CSV 파일을 다운로드한 후 [Close]를 클릭합니다.
3. 로컬 시스템의 다음 위치에 있는 AWS 자격 증명 프로필 파일에서 자격 증명을 설정합니다.
 - Linux, macOS 또는 Unix의 경우 `~/.aws/credentials`
 - Windows의 경우 `C:\Users\USERNAME\.aws\credentials`

이 파일에는 다음 형식의 행이 포함되어야 합니다.

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

`your_access_key_id` 및 `your_secret_access_key`를 내 액세스 키 ID와 보안 액세스 키로 대체합니다.

4. 로컬 시스템의 다음 위치에 있는 AWS Config 파일에서 기본 AWS 리전을 설정합니다.
 - Linux, macOS 또는 Unix의 경우 `~/.aws/config`
 - Windows의 경우 `C:\Users\USERNAME\.aws\config`

이 파일에는 다음 행이 포함되어야 합니다.

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

`your_aws_region`을 원하는 AWS 리전(예를 들면 "us-west-2")으로 대체합니다.

Note

리전을 선택하지 않으면 기본적으로 us-east-1이 사용됩니다.

다음 단계

[3단계: AWS CLI 및 AWS SDK API를 사용하여 시작하기 \(p. 13\)](#)

3단계: AWS CLI 및 AWS SDK API를 사용하여 시작하기

사용할 AWS CLI와 AWS SDK를 설정했으면 Amazon Rekognition을 사용하는 애플리케이션을 빌드할 수 있습니다. 다음 주제에서는 Amazon Rekognition Image 및 Amazon Rekognition Video를 시작하는 방법을 보여 줍니다.

- [이미지 작업 \(p. 35\)](#)
- [저장된 비디오 작업 \(p. 57\)](#)
- [스트리밍 비디오 작업 \(p. 84\)](#)

AWS CLI 예제 형식

이 가이드의 AWS CLI 예제는 Linux 운영 체제용입니다. Microsoft Windows에서 샘플을 사용하려면 --image 파라미터의 JSON 형식을 변경하고 줄 바꿈을 백슬래시(\)에서 캐럿 기호(^)로 변경해야 합니다. JSON 형식에 대한 자세한 내용은 [AWS 명령줄 인터페이스 파라미터 값 지정](#)을 참조하십시오. 다음은 Microsoft Windows용 AWS CLI 명령의 예입니다.

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

Microsoft Windows와 Linux에서 모두 작동하는 JSON 간편 버전을 제공할 수도 있습니다.

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

자세한 내용은 [AWS 명령줄 인터페이스에서 간편 구문 사용](#)을 참조하십시오.

다음 단계

[4단계: Amazon Rekognition 콘솔 사용 시작하기 \(p. 13\)](#)

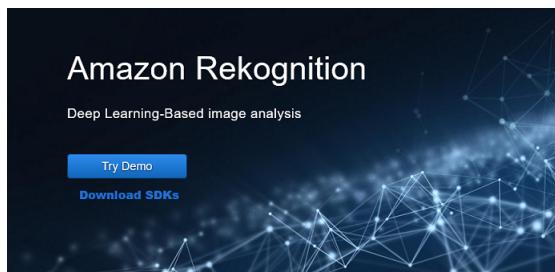
4단계: Amazon Rekognition 콘솔 사용 시작하기

이 단원은 이미지 집합에서 객체 및 장면 감지, 얼굴 분석, 얼굴 비교 같은 Amazon Rekognition 기능의 하위 집합을 사용하는 방법을 보여 줍니다. 자세한 내용은 [Amazon Rekognition: 작동 방식 \(p. 4\)](#) 단원을 참조하십시오. 또한 Amazon Rekognition API 또는 AWS CLI를 사용하여 객체, 장면, 얼굴을 감지하고 얼굴 비교 및 감지를 할 수 있습니다. 자세한 내용은 [3단계: AWS CLI 및 AWS SDK API를 사용하여 시작하기 \(p. 13\)](#) 단원을 참조하십시오.

이 단원은 Rekognition 콘솔을 사용하여 Rekognition의 Amazon CloudWatch 측정치 집계를 보는 방법도 보여 줍니다.

[항목](#)

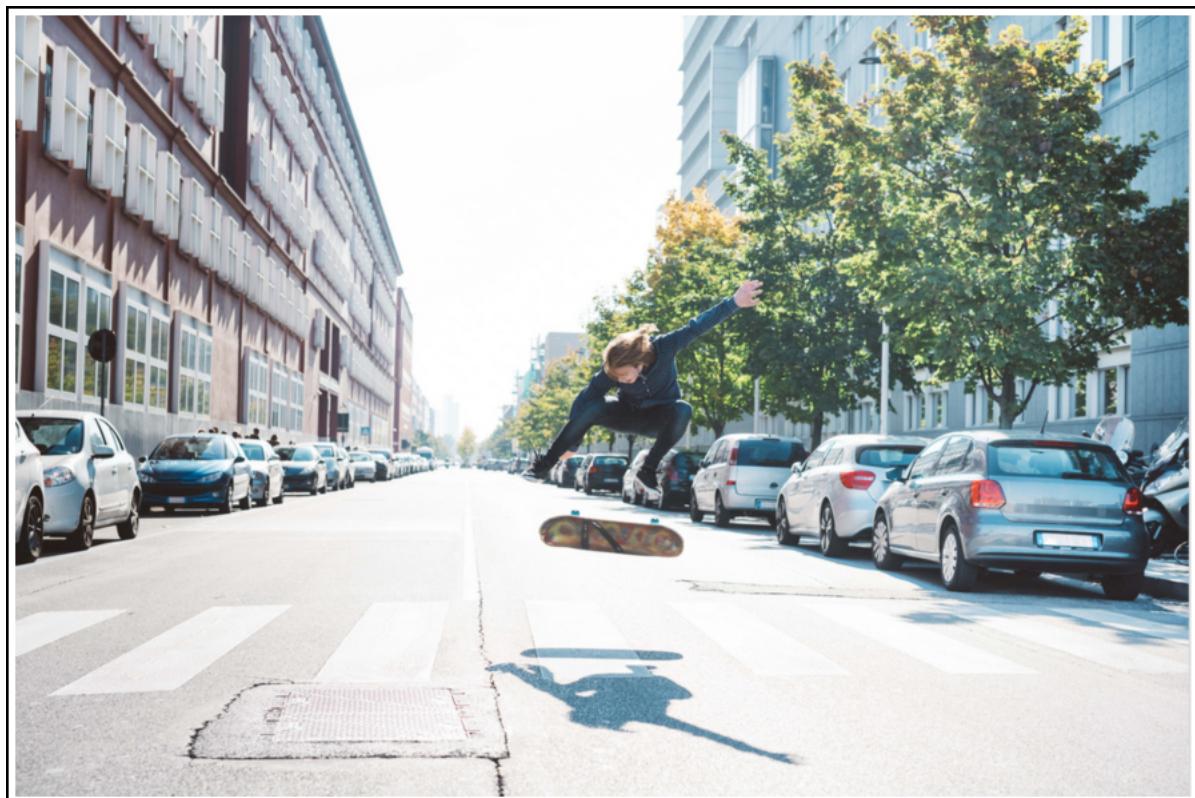
- 연습 1: 이미지에서 객체 및 장면 감지(콘솔) (p. 14)
- 연습 2: 이미지 내 얼굴 분석(콘솔) (p. 18)
- 연습 3: 이미지 내 얼굴 비교(콘솔) (p. 21)
- 연습 4: 집계 측정치 보기(콘솔) (p. 23)



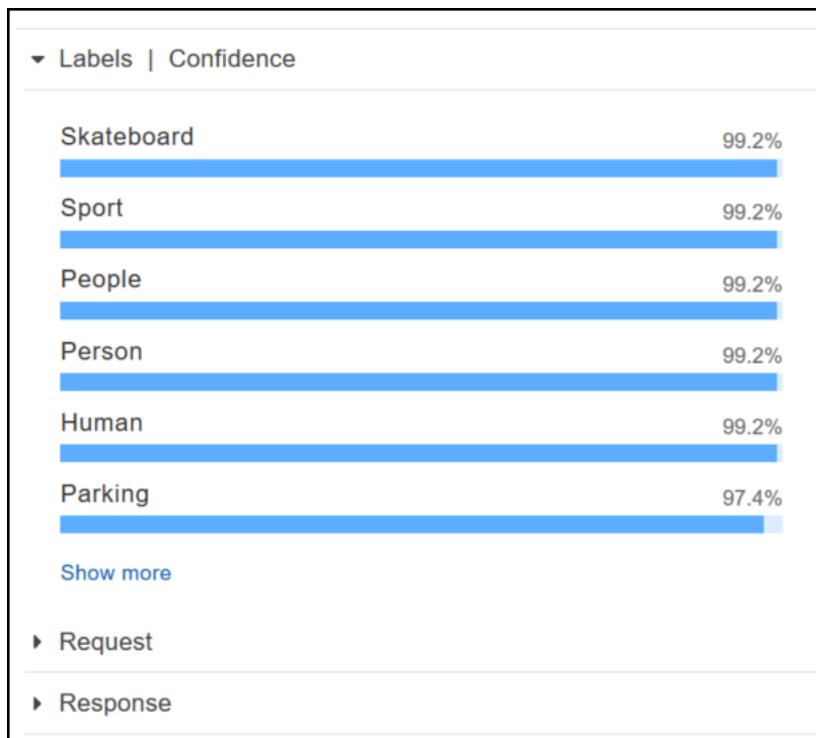
연습 1: 이미지에서 객체 및 장면 감지(콘솔)

이 단원은 Amazon Rekognition의 객체 및 장면 감지 기능의 작동 방식을 개요 수준에서 보여 줍니다. 이미지를 입력으로 지정하면 서비스는 해당 이미지에서 객체와 장면을 감지하고 각 객체와 장면의 백분율 신뢰도 점수와 함께 객체 및 장면을 반환합니다.

예를 들어 Amazon Rekognition은 샘플 이미지에서 객체와 장면(스케이트보드, 스포츠, 사람, 자동차, 차량)을 감지합니다. 이 응답에 표시된 모든 신뢰도 점수를 보려면 [Labels | Confidence] 창에서 [Show more]를 선택합니다.



Amazon Rekognition은 다음 샘플 응답에 나온 것처럼 샘플 이미지에서 감지된 각 객체의 신뢰도 점수도 반환합니다.



API에 대한 요청과 API의 응답을 참조로 볼 수도 있습니다.

요청

```
{  
    "contentString":{  
        "Attributes": [  
            "ALL"  
        ],  
        "Image": {  
            "S3Object": {  
                "Bucket": "console-sample-images",  
                "Name": "skateboard.jpg"  
            }  
        }  
    }  
}
```

응답

```
{  
    "Labels": [  
        {  
            "Confidence": 99.25359344482422,  
            "Name": "Skateboard"  
        },  
        {  
            "Confidence": 99.25359344482422,  
            "Name": "Sport"  
        },  
        {  
            "Confidence": 99.24723052978516,  
            "Name": "People"  
        }  
    ]  
}
```

```
},
{
    "Confidence":99.24723052978516,
    "Name":"Person"
},
{
    "Confidence":99.23908233642578,
    "Name":"Human"
},
{
    "Confidence":97.42484283447266,
    "Name":"Parking"
},
{
    "Confidence":97.42484283447266,
    "Name":"Parking Lot"
},
{
    "Confidence":91.53300476074219,
    "Name":"Automobile"
},
{
    "Confidence":91.53300476074219,
    "Name":"Car"
},
{
    "Confidence":91.53300476074219,
    "Name":"Vehicle"
},
{
    "Confidence":76.85114288330078,
    "Name":"Intersection"
},
{
    "Confidence":76.85114288330078,
    "Name":"Road"
},
{
    "Confidence":76.21503448486328,
    "Name":"Boardwalk"
},
{
    "Confidence":76.21503448486328,
    "Name":"Path"
},
{
    "Confidence":76.21503448486328,
    "Name":"Pavement"
},
{
    "Confidence":76.21503448486328,
    "Name":"Sidewalk"
},
{
    "Confidence":76.21503448486328,
    "Name":"Walkway"
},
{
    "Confidence":66.71541595458984,
    "Name":"Building"
},
{
    "Confidence":62.04711151123047,
    "Name":"Coupe"
},
{
```

```
        "Confidence":62.04711151123047,  
        "Name":"Sports Car"  
    },  
    {  
        "Confidence":61.98909378051758,  
        "Name":"City"  
    },  
    {  
        "Confidence":61.98909378051758,  
        "Name":"Downtown"  
    },  
    {  
        "Confidence":61.98909378051758,  
        "Name":"Urban"  
    },  
    {  
        "Confidence":60.978023529052734,  
        "Name":"Neighborhood"  
    },  
    {  
        "Confidence":60.978023529052734,  
        "Name":"Town"  
    },  
    {  
        "Confidence":59.22066116333008,  
        "Name":"Sedan"  
    },  
    {  
        "Confidence":56.48063278198242,  
        "Name":"Street"  
    },  
    {  
        "Confidence":54.235477447509766,  
        "Name":"Housing"  
    },  
    {  
        "Confidence":53.85226058959961,  
        "Name":"Metropolis"  
    },  
    {  
        "Confidence":52.001792907714844,  
        "Name":"Office Building"  
    },  
    {  
        "Confidence":51.325313568115234,  
        "Name":"Suv"  
    },  
    {  
        "Confidence":51.26075744628906,  
        "Name":"Apartment Building"  
    },  
    {  
        "Confidence":51.26075744628906,  
        "Name":"High Rise"  
    },  
    {  
        "Confidence":50.68067932128906,  
        "Name":"Pedestrian"  
    },  
    {  
        "Confidence":50.59548568725586,  
        "Name":"Freeway"  
    },  
    {  
        "Confidence":50.568580627441406,  
        "Name":"Bumper"
```

```
}
```

자세한 내용은 [Amazon Rekognition: 작동 방식 \(p. 4\)](#) 단원을 참조하십시오.

제공한 이미지에서 객체 및 장면 감지

Amazon Rekognition 콘솔에 소유하고 있는 이미지를 업로드하거나 이미지의 URL을 입력으로 제공할 수 있습니다. Amazon Rekognition은 제공된 이미지에서 감지한 객체와 장면 및 각 객체와 장면의 신뢰도 점수를 반환합니다.

Note

이미지는 크기가 5MB 미만이어야 하며 JPEG 또는 PNG 형식이어야 합니다.

제공한 이미지에서 객체 및 장면을 감지하려면

1. Amazon Rekognition 콘솔을 엽니다.
2. [Object and scene detection]을 선택합니다.
3. 다음 중 하나를 수행하십시오.
 - 이미지 업로드 - [Upload]를 선택하고 이미지를 저장한 위치로 이동한 다음 이미지를 선택합니다.
 - URL 사용 – 텍스트 상자에 URL을 입력한 다음 [Go]를 선택합니다.
4. [Labels | Confidence] 창에서 감지된 각 레이블의 신뢰도 점수를 확인합니다.

연습 2: 이미지 내 얼굴 분석(콘솔)

이 단원에서는 Amazon Rekognition 콘솔을 사용하여 이미지에서 얼굴을 감지하고 얼굴 속성을 분석하는 방법을 보여 줍니다. 얼굴이 포함된 이미지를 입력으로 제공하면 서비스는 이미지에서 얼굴을 감지하고 얼굴의 얼굴 속성을 분석한 다음 이미지에서 감지된 얼굴의 백분율 신뢰도 점수와 얼굴 속성을 반환합니다. 자세한 내용은 [Amazon Rekognition: 작동 방식 \(p. 4\)](#) 단원을 참조하십시오.

예를 들어 다음 샘플 이미지를 입력으로 선택하면 Amazon Rekognition은 이를 얼굴로 감지하고 감지된 얼굴의 신뢰도 점수와 얼굴 속성을 반환합니다.



다음은 샘플 응답을 보여 줍니다.

▼ 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

입력 이미지에 여러 얼굴이 있는 경우, Rekognition은 이미지에서 최대 100개의 얼굴을 감지합니다. 감지된 각 얼굴은 사각형으로 표시됩니다. 얼굴에서 사각형으로 표시된 영역을 클릭하면 Rekognition은 [Faces | Confidence] 창에 감지된 해당 얼굴의 신뢰도 점수와 얼굴 속성을 표시합니다.

제공한 이미지에서 얼굴 분석

Amazon Rekognition 콘솔에 자체 이미지를 업로드하거나 이미지 URL을 제공할 수 있습니다.

Note

이미지는 크기가 5MB 미만이어야 하며 JPEG 또는 PNG 형식이어야 합니다.

제공한 이미지의 얼굴을 분석하려면

1. Amazon Rekognition 콘솔을 엽니다.
2. [Facial analysis]를 선택합니다.

3. 다음 중 하나를 수행하십시오.
 - 이미지 업로드 - [Upload]를 선택하고 이미지를 저장한 위치로 이동한 다음 이미지를 선택합니다.
 - URL 사용 – 텍스트 상자에 URL을 입력한 다음 [Go]를 선택합니다.
4. [Faces | Confidence] 창에서 감지된 얼굴 중 하나의 신뢰도 점수와 얼굴 속성을 확인합니다.
5. 이미지에 여러 얼굴이 있는 경우, 다른 얼굴 중 하나를 선택해 속성과 점수를 봅니다.

연습 3: 이미지 내 얼굴 비교(콘솔)

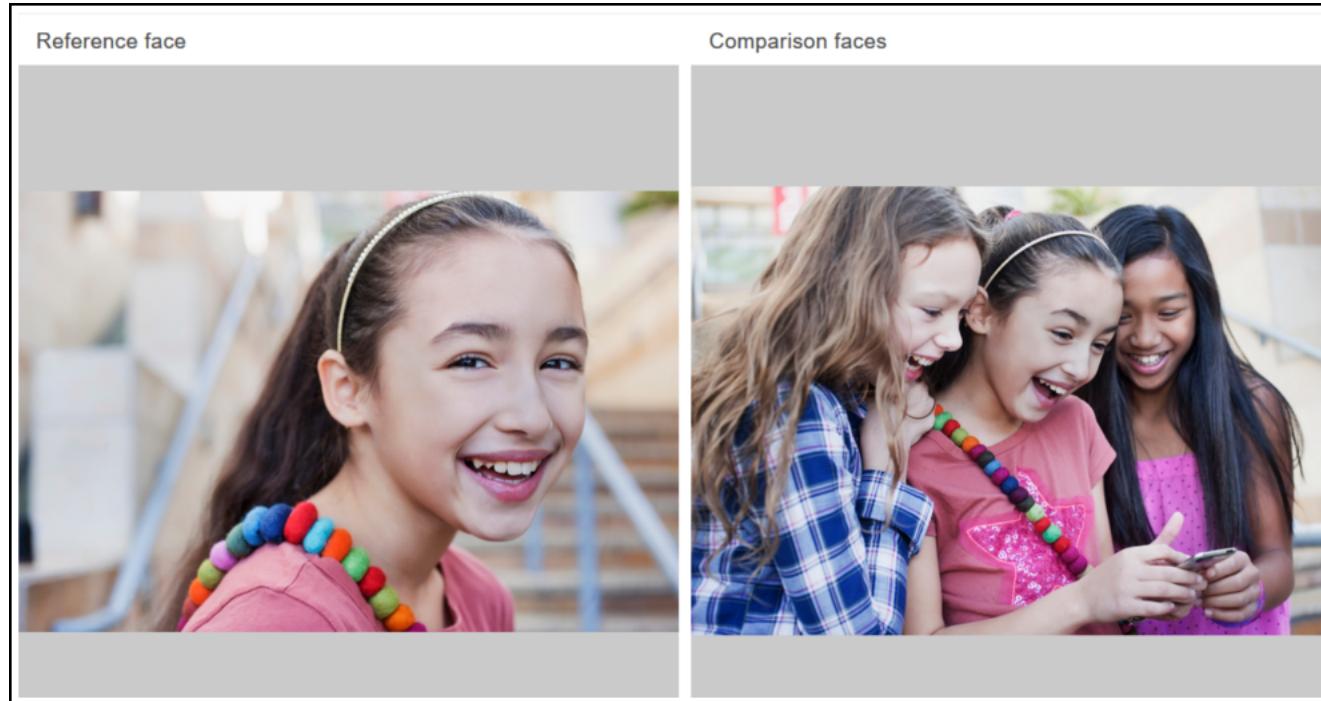
이 단원에서는 Amazon Rekognition 콘솔을 사용하여 여러 얼굴이 포함된 이미지 집합 내에서 얼굴을 비교하는 방법을 보여 줍니다. [Reference face](원본)와 [Comparison faces](대상) 이미지를 지정하면 Rekognition은 원본 이미지에서 가장 큰 얼굴(즉, 참조 얼굴)을 대상 이미지에서 감지된 최대 100개 얼굴(즉, 비교 얼굴)과 비교한 다음 원본의 얼굴이 대상 이미지의 얼굴과 얼마나 비슷한지 찾습니다. 각 비교의 유사성 점수는 [Results] 창에 표시됩니다.

대상 이미지에 여러 얼굴이 있는 경우, Rekognition은 원본 이미지의 얼굴을 대상 이미지의 최대 100개 얼굴과 비교한 다음 각 일치에 유사성 점수를 할당합니다.

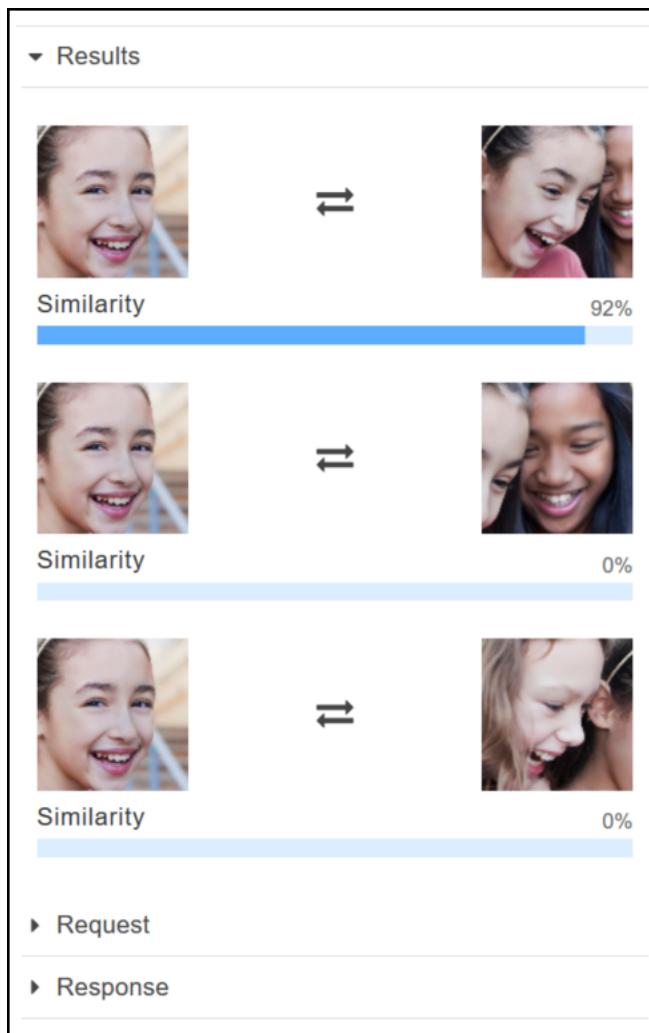
원본 이미지에 여러 얼굴이 포함되어 있는 경우, 서비스는 원본 이미지에서 가장 큰 얼굴을 감지하여 이를 사용해 대상 이미지에서 감지된 각각의 얼굴과 비교합니다.

자세한 내용은 [이미지에 있는 얼굴 비교](#) (p. 114) 단원을 참조하십시오.

예를 들어 왼쪽에 원본 이미지로 표시된 샘플 이미지와 오른쪽에 대상 이미지로 표시된 샘플 이미지의 경우, Rekognition은 원본 이미지의 얼굴과 대상 이미지의 각각의 얼굴을 비교하고 일치 여부를 확인한 다음 감지한 각 얼굴의 유사성 점수를 표시합니다.



다음은 대상 이미지에서 감지된 얼굴들과 각 얼굴의 유사성 점수를 보여 줍니다.



제공한 이미지 내 얼굴 비교

자체 원본 이미지와 대상 이미지를 업로드하면 Rekognition이 이미지에서 얼굴을 비교할 수 있습니다. 또는 이미지 위치의 URL을 지정할 수도 있습니다.

Note

이미지는 크기가 5MB 미만이어야 하며 JPEG 또는 PNG 형식이어야 합니다.

이미지에서 얼굴을 비교하려면

1. Amazon Rekognition 콘솔을 엽니다.
2. [Face comparison]을 선택합니다.
3. 원본 이미지에 대해 다음 중 하나를 수행합니다.
 - 이미지 업로드 – 왼쪽에서 [Upload]를 선택하고 원본 이미지를 저장한 위치로 이동한 다음 이미지를 선택합니다.
 - URL 사용 – 텍스트 상자에 원본 이미지의 URL을 입력한 다음 [Go]를 선택합니다.
4. 대상 이미지에 대해 다음 중 하나를 수행합니다.

- 이미지 업로드 – 오른쪽에서 [Upload]를 선택하고 원본 이미지를 저장한 위치로 이동한 다음 이미지를 선택합니다.
 - URL 사용 – 텍스트 상자에 원본 이미지의 URL을 입력한 다음 [Go]를 선택합니다.
5. Rekognition은 원본 이미지에서 가장 큰 얼굴을 대상 이미지의 최대 100개 얼굴과 비교한 다음 [Results] 창에 각 쌍의 유사성 점수를 표시합니다.

연습 4: 집계 측정치 보기(콘솔)

Amazon Rekognition 측정치 창은 지정된 기간 동안의 개별 Rekognition 측정치 집계를 위한 활동 그래프를 표시합니다. 예를 들어 SuccessfulRequestCount 집계 측정치는 지난 7일 동안 모든 Rekognition API 작업에 대한 성공적 요청 총수를 보여 줍니다.

다음 표에는 Rekognition 측정치 창에 표시되는 그래프와 그에 해당하는 Rekognition 측정치가 나열되어 있습니다. 자세한 내용은 [Rekognition의 CloudWatch 측정치 \(p. 210\)](#) 단원을 참조하십시오.

그래프	집계된 측정치
성공적 호출	SuccessfulRequestCount
클라이언트 오류	UserErrorCount
서버 오류	ServerErrorCount
병목 현상 발생	ThrottledCount
감지된 레이블	DetectedLabelCount
감지된 얼굴	DetectedFaceCount

각 그래프는 지정된 기간 동안 수집된 집계 측정치 데이터를 보여 줍니다. 해당 기간 동안 집계된 측정치 데이터의 총 개수도 표시됩니다. 개별 API 호출의 측정치를 보려면 각 그래프 아래의 링크를 선택합니다.

사용자가 Rekognition 측정치 창에 액세스하도록 허용하려면 해당 사용자에게 적절한 CloudWatch 및 Rekognition 권한이 있어야 합니다. 예를 들어 [AmazonRekognitionReadOnlyAccess](#) 및 [CloudWatchReadOnlyAccess](#) 관리형 정책 권한이 있는 사용자는 측정치 창을 볼 수 있습니다. 필요한 권한이 없는 사용자는 측정치 창을 열어도 그래프가 나타나지 않습니다. 자세한 내용은 [Amazon Rekognition에 대한 인증 및 액세스 제어 \(p. 24\)](#) 단원을 참조하십시오.

CloudWatch를 사용하여 Rekognition을 모니터링하는 방법에 대한 자세한 내용은 [모니터링 \(p. 207\)](#)을 참조하십시오.

집계 측정치를 보려면(콘솔)

1. <https://console.aws.amazon.com/rekognition/>에서 Amazon Rekognition 콘솔을 엽니다.
2. 탐색 창에서 [Metrics]를 선택합니다.
3. 드롭다운 목록에서 원하는 측정치 기간을 선택합니다.
4. 그래프를 업데이트 하려면 [Refresh] 버튼을 선택합니다.
5. 특정 집계 측정치의 상세한 CloudWatch 측정치를 보려면 측정치 그래프 아래의 [See details on CloudWatch]를 선택합니다.

Amazon Rekognition에 대한 인증 및 액세스 제어

Amazon Rekognition에 액세스하려면 자격 증명이 필요합니다. 이 자격 증명에는 Amazon Rekognition collection과 같은 AWS 리소스에 액세스할 수 있는 권한이 있어야 합니다. 다음 단원에서는 리소스 액세스를 보호하기 위해 [AWS Identity and Access Management\(IAM\)](#) 및 Amazon Rekognition 사용법에 대한 세부 정보를 제공합니다.

- [인증 \(p. 24\)](#)
- [액세스 제어 \(p. 25\)](#)

인증

다음과 같은 자격 증명 유형으로 AWS에 액세스할 수 있습니다.

- AWS 계정 루트 사용자 – AWS 계정을 처음 생성하는 경우에는 전체 AWS 서비스 및 계정 리소스에 대해 완전한 액세스 권한을 지닌 단일 로그인 자격 증명으로 시작합니다. 이 자격 증명은 AWS 계정 루트 사용자라고 하며, 계정을 생성할 때 사용한 이메일 주소와 암호로 로그인하여 액세스합니다. 일상적인 작업, 심지어 관리 작업의 경우에도 루트 사용자를 사용하지 마실 것을 강력히 권장합니다. 대신, [IAM 사용자를 처음 생성할 때만 루트 사용자를 사용하는 모범 사례](#)를 준수합니다. 그런 다음 루트 사용자를 안전하게 보관해 두고 몇 가지 계정 및 서비스 관리 작업을 수행할 때만 자격 증명을 사용합니다.
- IAM 사용자 – [IAM 사용자](#)는 특정 사용자 지정 권한(예: Amazon Rekognition에서 a collection을 만들 권한)이 있는 AWS 계정 내의 자격 증명입니다. IAM 사용자 이름과 암호를 사용하여 [AWS Management 콘솔](#), [AWS 토큰 포럼](#) 또는 [AWS Support Center](#)와 같은 보안 AWS 웹 페이지에 로그인할 수 있습니다.

사용자 이름과 암호 외에도 각 사용자에 대해 [액세스 키](#)를 생성할 수 있습니다. [여러 SDK 중 하나](#)를 통해 또는 [AWS Command Line Interface\(CLI\)](#)를 사용하여 AWS 서비스에 프로그래밍 방식으로 액세스할 때 이러한 키를 사용할 수 있습니다. SDK 및 CLI 도구는 액세스 키를 사용하여 암호화 방식으로 요청에 서명합니다. AWS 도구를 사용하지 않는 경우 요청에 직접 서명해야 합니다. Amazon Rekognition supports는 인바운드 API 요청을 인증하기 위한 프로토콜인 서명 버전 4를 지원합니다. 요청 인증에 대한 자세한 내용은 [AWS General Reference](#)의 [서명 버전 4 서명 프로세스](#)를 참조하십시오.

- IAM 역할 - [IAM 역할](#)은 특정 권한을 가진 계정에 생성할 수 있는 IAM 자격 증명입니다. IAM 사용자와 유사하지만, 특정 개인과 연결되지 않습니다. IAM 역할을 사용하면 AWS 서비스 및 리소스에 액세스하는데 사용할 수 있는 임시 액세스 키를 얻을 수 있습니다. 임시 자격 증명이 있는 IAM 역할은 다음과 같은 상황에서 유용합니다.
 - 연합된 사용자 액세스 – IAM 사용자를 만드는 대신 AWS Directory Service의 기존 사용자 자격 증명, 엔터프라이즈 사용자 디렉터리 또는 웹 자격 증명 공급자를 사용할 수 있습니다. 이러한 사용자를 연합된 사용자라고 합니다. [자격 증명 공급자](#)를 통해 액세스를 요청하면 AWS가 연합된 사용자에게 역할을 할당합니다. 연합된 사용자에 대한 자세한 내용은 IAM 사용 설명서의 [연합된 사용자 및 역할](#)을 참조하십시오.
- AWS 서비스 액세스 – 계정의 IAM 역할을 사용하여 AWS 서비스에 계정의 리소스에 액세스할 수 있는 권한을 부여할 수 있습니다. 예를 들어 Amazon Redshift에서 사용자 대신 Amazon S3 버킷에 액세스하도록 허용하는 역할을 만든 후 그 버킷으로부터 데이터를 Amazon Redshift 클러스터로 로드할 수 있습

니다. 자세한 내용은 [IAM 사용 설명서](#)의 Creating a Role to Delegate Permissions to an AWS Service 단원을 참조하십시오.

- Amazon EC2에서 실행하는 애플리케이션 – EC2 인스턴스에서 실행되고 AWS API 요청을 하는 애플리케이션의 임시 자격 증명을 IAM 역할을 사용하여 관리할 수 있습니다. 이것은 EC2 인스턴스 내에 액세스 키를 저장하는 경우에 바람직한 방법입니다. EC2 인스턴스에 AWS 역할을 할당하고 그 역할을 모든 애플리케이션에서 사용할 수 있도록 하려면 인스턴스에 연결된 인스턴스 프로파일을 만들어야 합니다. 인스턴스 프로파일에는 역할이 포함되어 있으며 EC2 인스턴스에서 실행되는 프로그램이 임시 자격 증명을 얻을 수 있습니다. 자세한 내용은 IAM 사용 설명서의 [IAM 역할을 사용하여 Amazon EC2 인스턴스에서 실행되는 애플리케이션에 권한 부여하기](#) 섹션을 참조하십시오.

액세스 제어

요청을 인증하는 데 유효한 자격 증명이 있더라도 권한이 없다면 Amazon Rekognition 리소스를 생성하거나 액세스할 수 없습니다. 예를 들어 Amazon Rekognition collection을 생성할 권한이 있어야 합니다.

다음 단원에서는 Amazon Rekognition에 대한 권한을 관리하는 방법에 대해 설명합니다. 먼저 개요를 읽어 보면 도움이 됩니다.

- [Amazon Rekognition 리소스에 대한 액세스 권한 관리 개요](#) (p. 25)
- [Amazon Rekognition에 대한 자격 증명 기반 정책\(IAM 정책\) 사용](#) (p. 28)
- [Amazon Rekognition API 권한: 작업, 권한 및 리소스 참조](#) (p. 31)

Amazon Rekognition 리소스에 대한 액세스 권한 관리 개요

모든 AWS 리소스는 AWS 계정의 소유이고, 리소스 생성 또는 리소스 액세스 권한은 권한 정책에 따라 결정됩니다. 계정 관리자는 IAM 자격 증명(즉, 사용자, 그룹, 역할)에 권한 정책을 연결할 수 있고, 일부 서비스(예: AWS Lambda)에서는 리소스에 대한 권한 정책 연결도 지원합니다.

Note

계정 관리자 또는 관리자 사용자는 관리자 권한이 있는 사용자입니다. 자세한 정보는 IAM 사용 설명서에서 [IAM 모범 사례](#)를 참조하십시오.

권한을 부여하려면 권한을 부여 받을 사용자, 권한 대상이 되는 리소스, 해당 리소스에 허용되는 특정 작업을 결정합니다.

항목

- [Amazon Rekognition 리소스 및 작업](#) (p. 25)
- [리소스 소유권 이해](#) (p. 26)
- [리소스 액세스 관리](#) (p. 26)
- [정책 요소 지정: 작업, 효과, 보안 주체](#) (p. 28)
- [정책에서 조건 지정](#) (p. 28)

Amazon Rekognition 리소스 및 작업

Amazon Rekognition에는 a collection 및 스트림 프로세서에 대한 여러 리소스 유형이 있습니다. 정책에서 Amazon Resource Name(ARN)을 사용하여 정책이 적용되는 리소스를 식별합니다.

다음 표에 나와 있는 것처럼 이러한 리소스에는 고유한 Amazon Resource Name(ARN)이 연결됩니다.

리소스 유형	ARN 형식
모음 ARN	<code>arn:aws:rekognition:region:account-id:collection/ collection-id</code>
스트림 프로세서 ARN	<code>arn:aws:rekognition:region:account-id:streamprocessor/ stream-processor-name</code>

Amazon Rekognition은(는) Amazon Rekognition 리소스를 처리하기 위한 작업을 제공합니다. 사용 가능한 작업 목록은 Amazon Rekognition [Amazon Rekognition API 권한: 작업, 권한 및 리소스 참조 \(p. 31\)](#)를 참조하십시오.

리소스 소유권 이해

AWS 계정은 리소스를 누가 생성했든 상관없이 계정에서 생성된 리소스를 소유합니다. 구체적으로 리소스 소유자는 리소스 생성 요청을 인증하는 [보안 주체 엔터티](#)(즉, 루트 계정 또는 IAM 사용자)의 AWS 계정입니다. 다음 예에서는 이 계정의 작동 방식을 설명합니다.

- AWS 계정의 루트 계정 자격 증명을 사용하여 a collection를 만들면 AWS 계정은 해당 리소스의 소유자가 됩니다(Amazon Rekognition에서 리소스는 a collection).
- AWS 계정에서 IAM 사용자를 만들고 a collection를 생성할 수 있는 권한을 사용자에게 부여하면 해당 사용자가 a collection을 생성할 수 있습니다. 하지만 collection 리소스는 해당 사용자가 속한 AWS 계정이 소유합니다.

리소스 액세스 관리

권한 정책은 누가 무엇에 액세스 할 수 있는지를 나타냅니다. 다음 단원에서는 권한 정책을 만드는데 사용 가능한 옵션에 대해 설명합니다.

Note

이 단원에서는 Amazon Rekognition의 맥락에서 IAM을 사용하는 방법에 대해 설명하며, IAM 서비스에 대한 자세한 정보는 다루지 않습니다. 전체 IAM 설명서는 [IAM이란 무엇인가?](#)(출처: IAM 사용 설명서) 단원을 참조하십시오. IAM 정책 구문과 설명에 대한 자세한 정보는 IAM 사용 설명서의 [AWS IAM Policy Reference](#)를 참조하십시오.

IAM 자격 증명에 연결된 정책을 자격 증명 기반 정책(IAM 정책)이라고 하고, 리소스에 연결된 정책을 리소스 기반 정책이라고 합니다. Amazon Rekognition은 자격 증명 기반 정책을 지원합니다.

항목

- [자격 증명 기반 정책\(IAM 정책\) \(p. 26\)](#)
- [리소스 기반 정책 \(p. 27\)](#)

자격 증명 기반 정책(IAM 정책)

정책을 IAM 자격 증명에 연결할 수 있습니다. 예를 들면,

- 계정 내 사용자 또는 그룹에 권한 정책 연결 – a collection과 같은 Amazon Rekognition 리소스를 생성할 사용자 권한을 부여하기 위해 권한 정책을 특정 사용자 또는 해당 사용자가 속한 그룹에 연결할 수 있습니다.
- 역할에 권한 정책 연결(교차 계정 권한 부여) – 자격 증명 기반 권한 정책을 IAM 역할에 연결하여 교차 계정 권한을 부여할 수 있습니다. 예를 들어, 계정 A의 관리자는 다음과 같이 다른 AWS 계정(예: 계정 B) 또는 AWS 서비스에 교차 계정 권한을 부여할 역할을 생성할 수 있습니다.
 1. 계정 A 관리자는 IAM 역할을 생성하고 계정 A의 리소스에 대한 권한을 부여하는 역할에 권한 정책을 연결합니다.

2. 계정 A 관리자는 계정 B를 역할을 수임할 보안 주체로 식별하는 역할에 신뢰 정책을 연결합니다.
3. 계정 B 관리자는 계정 B의 사용자에게 역할을 수임할 권한을 위임할 수 있습니다. 그러면 계정 B의 사용자가 계정 A에서 리소스를 생성하거나 액세스할 수 있습니다. AWS 서비스에 역할 수임 권한을 부여 할 경우 신뢰 정책의 보안 주체가 AWS 서비스 보안 주체이기도 합니다.

IAM을 사용하여 권한을 위임하는 방법에 대한 자세한 정보는 [IAM 사용 설명서의 액세스 관리를 참조하십시오.](#)

다음은 모든 모음을 나열하는 정책 예제입니다.

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

Amazon Rekognition에서 자격 증명 기반 정책을 사용하는 방법에 대한 자세한 내용은 [Amazon Rekognition에 대한 자격 증명 기반 정책\(IAM 정책\) 사용 \(p. 28\)](#) 단원을 참조하십시오. 사용자, 그룹, 역할 및 권한에 대한 자세한 내용은 IAM 사용 설명서의 [자격 증명\(사용자, 그룹 및 역할\)](#) 단원을 참조하십시오.

리소스 기반 정책

Amazon S3와 같은 다른 서비스도 리소스 기반 권한 정책을 지원합니다. 예를 들어, 정책을 S3 버킷에 연결하여 해당 버킷에 대한 액세스 권한을 관리할 수 있습니다. Amazon Rekognition은(는) 리소스 기반 정책을 지원하지 않습니다.

Amazon S3 버킷에 저장된 이미지에 액세스하려면 S3 버킷에 있는 객체에 대한 액세스 권한이 있어야 합니다. 이 권한이 있으면 Amazon Rekognition이 S3 버킷에서 이미지를 다운로드할 수 있습니다. 다음 정책 예제에서 사용자는 Tests3bucket이라는 S3 버킷에서 s3:GetObject 작업을 수행할 수 있습니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "s3:GetObject",
            "Resource": [
                "arn:aws:s3:::Tests3bucket"
            ]
        }
    ]
}
```

버전 관리를 활성화한 상태로 S3 버킷을 사용하려면 다음 예제와 같이 s3:GetObjectVersion 작업을 추가합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "s3:GetObjectVersion"
        }
    ]
}
```

```
    "Action": [
        "s3:GetObject",
        "s3:GetObjectVersion"
    ],
    "Resource": [
        "arn:aws:s3:::Tests3bucket"
    ]
}
```

정책 요소 지정: 작업, 효과, 보안 주체

각 Amazon Rekognition 리소스에 대해 서비스는 일련의 API 작업을 정의합니다. 이러한 API 작업에 대한 권한을 부여하기 위해 Amazon Rekognition에서는 정책에서 지정할 수 있는 작업을 정의합니다. 일부 API 작업에서는 API 작업을 수행하기 위해 복수의 작업에 대한 권한이 필요할 수 있습니다. 리소스 및 API 작업에 대한 자세한 내용은 [Amazon Rekognition 리소스 및 작업 \(p. 25\)](#) 및 [Amazon Rekognition API 권한: 작업, 권한 및 리소스 참조 \(p. 31\)](#)를 참조하십시오.

다음은 가장 기본적인 정책 요소입니다.

- Resource – Amazon Resource Name(ARN)을 사용하여 정책을 적용할 리소스를 식별합니다. 자세한 내용은 [Amazon Rekognition 리소스 및 작업 \(p. 25\)](#) 단원을 참조하십시오.
- 작업 – 작업 키워드를 사용하여 허용 또는 거부할 리소스 작업을 식별합니다. 예를 들어 `ListCollections`를 사용하여 모음을 나열할 수 있습니다.
- Effect – 사용자가 특정 작업을 요청하는 경우의 결과를 지정합니다. 명시적으로 리소스에 대한 액세스 권한을 부여(허용)하지 않는 경우, 액세스는 묵시적으로 거부됩니다. 다른 정책에서는 액세스 권한을 부여하더라도 리소스에 대한 액세스를 명시적으로 거부하여 사용자가 해당 리소스에 액세스하지 못하게 할 수도 있습니다.
- 보안 주체 – 자격 증명 기반 정책(IAM 정책)에서, 해당 정책이 연결된 사용자를 묵시적인 보안 주체라고 합니다. 리소스 기반 정책의 경우 사용자, 계정, 서비스 또는 권한의 수신자인 기타 엔터티를 지정합니다(리소스 기반 정책에만 해당). Amazon Rekognition은(는) 리소스 기반 정책을 지원하지 않습니다.

IAM 정책 구문과 설명에 대한 자세한 내용은 IAM 사용 설명서의 [AWS IAM 정책 참조](#) 단원을 참조하십시오.

모든 Amazon Rekognition API 작업과 해당 작업이 적용되는 리소스를 보여주는 목록은 [Amazon Rekognition API 권한: 작업, 권한 및 리소스 참조 \(p. 31\)](#) 단원을 참조하십시오.

정책에서 조건 지정

권한을 부여할 때 액세스 정책 언어를 사용하여 조건이 적용되는 조건을 지정할 수 있습니다. 예를 들어, 특정 날짜 이후에만 정책을 적용할 수 있습니다. 정책 언어에서의 조건 지정에 관한 자세한 정보는 IAM 사용 설명서의 [조건](#)을 참조하십시오.

조건을 표시하려면 미리 정의된 조건 키를 사용합니다. Amazon Rekognition에 특정한 조건 키는 없습니다. 하지만 적절하게 사용할 수 있는 AWS 차원의 조건 키가 있습니다. AWS 차원 키의 전체 목록은 IAM 사용 설명서의 [사용 가능한 조건 키](#) 단원을 참조하십시오.

Amazon Rekognition에 대한 자격 증명 기반 정책 (IAM 정책) 사용

이 주제에서는 계정 관리자가 권한 정책을 IAM 자격 증명(즉, 사용자, 그룹 및 역할)에 연결하고 이 과정을 통해 Amazon Rekognition 리소스에서 작업을 수행할 권한을 부여할 수 있는 방법을 보여 주는 자격 증명 기반 정책의 예를 제공합니다.

Important

Amazon Rekognition 리소스에 대한 액세스 관리를 위해 제공되는 기본 개념과 옵션 설명에 대한 소개 주제 부분을 우선 읽어 보는 것이 좋습니다. 자세한 내용은 [Amazon Rekognition 리소스에 대한 액세스 권한 관리 개요 \(p. 25\)](#) 단원을 참조하십시오.

항목

- [Amazon Rekognition 콘솔 사용에 필요한 권한 \(p. 29\)](#)
- [Amazon Rekognition에 대한 AWS 관리형\(미리 정의된\) 정책 \(p. 29\)](#)
- [고객 관리형 정책 예 \(p. 30\)](#)

다음은 권한 정책의 예입니다.

```
"Version": "2012-10-17",
"Statement": [
    {
        "Effect": "Allow",
        "Action": [
            "rekognition:CompareFaces",
            "rekognition:DetectFaces",
            "rekognition:DetectLabels",
            "rekognition>ListCollections",
            "rekognition>ListFaces",
            "rekognition:SearchFaces",
            "rekognition:SearchFacesByImage"
        ],
        "Resource": "*"
    }
]
```

이 정책 예제에서는 제한된 Amazon Rekognition 작업 세트를 사용하여 사용자에게 리소스에 대한 읽기 전용 액세스를 부여합니다.

모든 Amazon Rekognition API 작업과 해당 작업이 적용되는 리소스를 보여주는 표는 [Amazon Rekognition API 권한: 작업, 권한 및 리소스 참조 \(p. 31\)](#)를 참조하십시오.

Amazon Rekognition 콘솔 사용에 필요한 권한

Amazon Rekognition 콘솔 작업 시 Amazon Rekognition에 추가 권한이 필요하지 않습니다.

Amazon Rekognition에 대한 AWS 관리형(미리 정의된) 정책

AWS는 AWS에서 생성하고 관리하는 독립형 IAM 정책을 제공하여 많은 일반 사용 사례를 처리합니다. 이러한 AWS 관리형 정책은 사용자가 필요한 권한을 조사할 필요가 없도록 일반 사용 사례에 필요한 권한을 부여합니다. 자세한 내용은 IAM 사용 설명서의 [AWS 관리형 정책](#)을 참조하십시오.

계정의 사용자에 연결할 수 있는 다음 AWS 관리형 정책은 Amazon Rekognition에 고유합니다.

- [AmazonRekognitionFullAccess](#) – 모음 생성 및 삭제를 포함하여 Amazon Rekognition 리소스에 대한 모든 액세스 권한을 부여합니다.
- [AmazonRekognitionReadOnlyAccess](#) – Amazon Rekognition 리소스에 대한 읽기 전용 액세스 권한을 부여합니다.
- [AmazonRekognitionServiceRole](#) – Amazon Rekognition은 Amazon Kinesis Data Streams 및 Amazon SNS 서비스를 자동으로 호출할 수 있습니다.

Note

IAM 콘솔에 로그인하고 이 콘솔에서 특정 정책을 검색하여 이러한 권한 정책을 검토할 수 있습니다. 이러한 정책은 AWS SDK 또는 AWS CLI를 사용하는 경우에 유효합니다.

Amazon Rekognition 작업 및 리소스에 대한 권한을 허용하는 고유의 사용자 지정 IAM 정책을 생성할 수도 있습니다. 해당 권한이 필요한 IAM 사용자 또는 그룹에 이러한 사용자 지정 정책을 연결할 수 있습니다.

고객 관리형 정책 예

이 단원에서는 다양한 Amazon Rekognition 작업에 대한 권한을 부여하는 사용자 정책의 예를 제공합니다. 이러한 정책은 AWS SDK 또는 AWS CLI를 사용하는 경우에 유효합니다. 콘솔을 사용하는 경우 [Amazon Rekognition 콘솔 사용에 필요한 권한](#) (p. 29)의 설명과 같이 콘솔에 특정한 추가 권한을 부여해야 합니다.

예

- [예제 1: 사용자에게 리소스에 대한 읽기 전용 액세스 허용](#) (p. 30)
- [예제 2: 사용자에게 리소스에 대한 모든 액세스 권한 허용](#) (p. 30)

예제 1: 사용자에게 리소스에 대한 읽기 전용 액세스 허용

다음 예제는 Amazon Rekognition 리소스에 대한 읽기 전용 액세스 권한을 부여합니다.

```
{  
    "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:DescribeStreamProcessor",  
                "rekognition>ListStreamProcessors"  
            ],  
            "Resource": "*"  
        }  
    ]  
}
```

예제 2: 사용자에게 리소스에 대한 모든 액세스 권한 허용

다음 예제는 Amazon Rekognition 리소스에 대한 모든 액세스 권한을 부여합니다.

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

Amazon Rekognition API 권한: 작업, 권한 및 리소스 참조

액세스 제어 (p. 25)을 설정하고 IAM 자격 증명에 연결할 수 있는 권한 정책(자격 증명 기반 정책)을 작성할 때 다음 목록을 참조로 사용할 수 있습니다. 이는 Amazon Rekognition API 작업, 부여된 권한으로 수행할 수 있는 작업, 권한 부여된 AWS 리소스가 각각 나열되어 있습니다. 정책의 Action 필드에서 작업을 지정하고, 정책의 Resource 필드에서 리소스 값을 지정합니다.

Amazon Rekognition 정책의 AWS 차원 조건 키를 사용하여 조건을 표시할 수 있습니다. AWS 차원 키의 전체 목록은 IAM 사용 설명서의 [사용할 수 있는 키](#)를 참조하십시오.

Note

작업을 지정하려면 rekognition 접두사 다음에 API 작업 이름을 사용합니다(예: rekognition:DeleteCollection).

Amazon Rekognition API 및 작업에 필요한 권한

API 작업: CreateCollection

필요한 권한(API 작업): rekognition:CreateCollection

리소스: arn:aws:rekognition:**region:account-id:collection/collection-id**

API 작업: DeleteCollection

필요한 권한(API 작업): rekognition:DeleteCollection

리소스: arn:aws:rekognition:**region:account-id:collection/collection-id**

API 작업: DeleteFaces

필요한 권한(API 작업): rekognition:DeleteFaces

리소스: arn:aws:rekognition:**region:account-id:collection/collection-id**

API 작업: DetectFaces

필요한 권한(API 작업): rekognition:DetectFaces

리소스: arn:aws:rekognition:**region:account-id:collection/collection-id**

API 작업: IndexFaces

필요한 권한(API 작업): rekognition:IndexFaces

리소스: arn:aws:rekognition:**region:account-id:collection/collection-id**

API 작업: ListCollections

필요한 권한(API 작업): rekognition:ListCollections

리소스: arn:aws:rekognition:**region:account-id:***

API 작업: ListFaces

필요한 권한(API 작업): rekognition>ListFaces

리소스: arn:aws:rekognition:**region:account-id:collection/collection-id**

API 작업: SearchFaces

필요한 권한(API 작업): rekognition:SearchFaces

리소스: arn:aws:rekognition:**region:account-id:collection/collection-id**

API 작업: SearchFacesByImage

필요한 권한(API 작업): rekognition:SearchFacesByImage

리소스: arn:aws:rekognition:**region:account-id:collection/collection-id**

API 작업: CreateStreamProcessor

필요한 권한(API 작업): rekognition>CreateStreamProcessor

리소스: arn:aws:rekognition:**region:account-id:collection/collection-id**

리소스: arn:aws:rekognition:**region:account-id:streamprocessor/stream-processor-name**

API 작업: DeleteStreamProcessor

필요한 권한(API 작업): rekognition>DeleteStreamProcessor

리소스: arn:aws:rekognition:**region:account-id:streamprocessor/stream-processor-name**

API 작업: ListStreamProcessors

필요한 권한(API 작업): rekognition>ListStreamProcessors

리소스: arn:aws:rekognition:**region:account-id:streamprocessor/stream-processor-name**

API 작업: StartStreamProcessor

필요한 권한(API 작업): rekognition:StartStreamProcessor

리소스: arn:aws:rekognition:**region:account-id:streamprocessor/stream-processor-name**

API 작업: StopStreamProcessor

필요한 권한(API 작업): rekognition:StopStreamProcessor

리소스: arn:aws:rekognition:**region:account-id:streamprocessor/stream-processor-name**

API 작업: CompareFaces

필요한 권한(API 작업): rekognition:CompareFaces

사용되지 않습니다.

API 작업: DetectFaces

필요한 권한(API 작업): rekognition:DetectFaces

사용되지 않습니다.

API 작업: DetectLabels

필요한 권한(API 작업): rekognition:DetectLabels

사용되지 않습니다.

API 작업: DetectModerationLabels

필요한 권한(API 작업): rekognition:DetectModerationLabels

사용되지 않습니다.

API 작업: DetectText

필요한 권한(API 작업): rekognition:DetectText

사용되지 않습니다.

API 작업: RecognizeCelebrities

필요한 권한(API 작업): rekognition:RecognizeCelebrities

사용되지 않습니다.

API 작업:

필요한 권한(API 작업): rekognition:

사용되지 않습니다.

API 작업: GetCelebrityRecognition

필요한 권한(API 작업): rekognition:GetCelebrityRecognition

사용되지 않습니다.

API 작업: GetContentModeration

필요한 권한(API 작업): rekognition:getContentModeration

사용되지 않습니다.

API 작업: GetFaceDetection

필요한 권한(API 작업): rekognition:GetFaceDetection

사용되지 않습니다.

API 작업: GetLabelDetection

필요한 권한(API 작업): rekognition:GetLabelDetection

사용되지 않습니다.

API 작업: GetPersonTracking

필요한 권한(API 작업): rekognition:getPersonTracking

사용되지 않습니다.

API 작업: StartCelebrityRecognition

필요한 권한(API 작업): rekognition:StartCelebrityRecognition

사용되지 않습니다.

API 작업: StartContentModeration

필요한 권한(API 작업): rekognition:StartContentModeration

사용되지 않습니다.

API 작업: StartFaceDetection

필요한 권한(API 작업): rekognition:StartFaceDetection

사용되지 않습니다.

API 작업: StartLabelDetection

필요한 권한(API 작업): rekognition:StartLabelDetection

사용되지 않습니다.

API 작업: StartPersonTracking

필요한 권한(API 작업): rekognition:StartPersonTracking

사용되지 않습니다.

이미지 작업

이 섹션에서는 Amazon Rekognition Image가 이미지에 대해 수행할 수 있는 감지 및 인식 유형에 대해 알아보겠습니다.

- 레이블 감지
- 얼굴 감지 및 비교
- 유명 인사 인식
- 이미지 조절
- 이미지 속 텍스트 감지

이러한 작업은 Amazon Rekognition Image가 작업에서 발견된 어떠한 정보도 유지하지 않는 비 스토리지 API 작업에 의해 수행됩니다. 입력 이미지 바이트는 비 스토리지 API 작업에 의해 유지되지 않습니다. 자세한 내용은 [비스토리지 및 스토리지 API 작업 \(p. 7\)](#) 단원을 참조하십시오.

또한 Amazon Rekognition Image는 나중에 검색할 수 있도록 얼굴 메타데이터를 모음에 저장할 수 있습니다. 자세한 내용은 [모음에서 얼굴 검색 \(p. 127\)](#) 단원을 참조하십시오.

이 섹션에서는 Amazon Rekognition Image API 작업을 사용하여 Amazon S3 버킷에 저장된 이미지와 로컬 파일 시스템에서 로드된 이미지 바이트를 분석합니다. 이 섹션에서는 .jpg 이미지에서 이미지 방향 정보를 가져오는 방법에 대해서도 알아봅니다.

항목

- [이미지 \(p. 35\)](#)
- [이미지 작업의 모범 사례 \(p. 36\)](#)
- [Amazon S3 버킷에 저장된 이미지 분석 \(p. 37\)](#)
- [로컬 파일 시스템에서 불러온 이미지 분석 \(p. 40\)](#)
- [이미지 방향 및 경계 상자 좌표 가져오기 \(p. 47\)](#)

이미지

Amazon Rekognition Image 작업은 .jpg 또는 .png 형식의 이미지를 분석할 수 있습니다.

이미지 바이트를 호출의 일부로 Amazon Rekognition Image 작업에 전달하거나 기존 Amazon S3 객체를 참조합니다. Amazon S3 버킷에 저장된 이미지를 분석하는 예제는 [Amazon S3 버킷에 저장된 이미지 분석 \(p. 37\)](#)을 참조하십시오. 이미지 바이트를 Amazon Rekognition Image API 작업으로 전달하는 예제는 [로컬 파일 시스템에서 불러온 이미지 분석 \(p. 40\)](#) 단원을 참조하십시오.

HTTP를 사용하여 Amazon Rekognition Image 작업의 일부로 이미지 바이트를 전달하는 경우 이미지 바이트는 base64로 인코딩된 문자열이어야 합니다. AWS SDK를 사용하여 API 작업 호출의 일부로 이미지 바이트를 전달하는 경우 이미지 바이트를 base64로 인코딩해야 하는지 여부는 사용하는 언어에 따라 달라집니다.

다음의 일반적인 AWS SDK는 이미지를 base64로 자동 인코딩하므로 Amazon Rekognition Image API 작업을 호출하기 전에 이미지 바이트를 인코딩할 필요가 없습니다.

- Java
- JavaScript
- Python
- PHP

다른 AWS SDK를 사용하는 경우 Rekognition API 작업을 호출할 때 이미지 형식 오류가 발생하면 이미지 바이트를 Rekognition API 작업에 전달하기 전에 base64로 인코딩하십시오.

AWS CLI를 사용하여 Amazon Rekognition Image 작업을 호출하는 경우 호출의 일부로 이미지 바이트를 전달하는 작업은 지원되지 않습니다. 먼저 이미지를 Amazon S3 버킷에 업로드한 다음 업로드된 이미지를 참조하는 작업을 호출해야 합니다.

Note

이미지 바이트 대신 S3Object에 저장된 이미지를 전달하면 이미지를 base64로 인코딩할 필요가 없습니다.

Amazon Rekognition Image 작업에 대한 지연 시간을 최소화하는 방법에 대한 자세한 내용은 [Amazon Rekognition Image 작업 지연 시간 \(p. 36\)](#) 단원을 참조하십시오.

이미지 방향 설정

여러 Rekognition API 작업에서는 분석된 이미지의 방향이 반환됩니다. 예를 들어 [DetectFaces \(p. 243\)](#) API 작업은 OrientationCorrection 필드에 원본 이미지에 대한 방향 정보를 반환합니다. 이미지 방향을 알면 이미지 방향을 전환하여 표시할 수 있으므로 중요합니다. 얼굴을 분석하는 Rekognition API 작업은 이미지 속의 얼굴 위치에 대한 경계 상자도 반환합니다. 경계 상자를 사용하여 이미지의 얼굴 주위에 상자를 표시할 수 있습니다. 반환된 경계 상자 좌표는 이미지 방향의 영향을 받으며 얼굴 주위에 상자를 올바르게 표시하려면 경계 상자 좌표를 변환해야 할 수도 있습니다. 자세한 내용은 [이미지 방향 및 경계 상자 좌표 가져오기 \(p. 47\)](#) 단원을 참조하십시오.

이미지 작업의 모범 사례

다음은 이미지 작업의 모범 사례입니다.

Amazon Rekognition Image 작업 지연 시간

Amazon Rekognition Image 작업에 대한 지연 시간을 최소화하려면 다음을 고려하십시오.

- 이미지가 포함된 Amazon S3 버킷의 리전과 Amazon Rekognition Image API 작업에 사용하는 리전이 일치해야 합니다.
- 이미지 바이트로 Amazon Rekognition Image 작업을 호출하는 것이 이미지를 Amazon S3 버킷에 업로드한 다음 Amazon Rekognition Image 작업에서 업로드된 이미지를 참조하는 것보다 더 빠릅니다. 거의 실시간으로 처리하기 위해 이미지를 Amazon Rekognition Image에 업로드하는 경우 이 접근 방식을 고려하십시오. 예를 들어 IP 카메라에서 업로드한 이미지 또는 웹 포털을 통해 업로드한 이미지입니다.
- 이미지가 Amazon S3 버킷에 이미 있는 경우 Amazon Rekognition Image 작업에서 해당 이미지를 참조하는 것이 이미지 바이트를 작업에 전달하는 것보다 더 빠를 것입니다.
- 저해상도 이미지는 고해상도 이미지보다 더 빠르게 처리됩니다. 더 빠른 처리를 위해 입력 이미지를 더 낮은 해상도로 조정하십시오.

얼굴 인식 입력 이미지에 대한 권장 사항

Amazon Rekognition Image는 다양한 이미지 조건 및 크기에서 작동하지만 얼굴 인식을 위한 참조 사진을 선택할 때는 다음 지침에 따르는 것이 좋습니다.

- 정면을 향한 얼굴이 있어야 합니다.
- 그림자 등 음영이 바뀌지 않도록 평면 조명이 얼굴을 비춰야 합니다.
- 배경과 충분한 대비를 이뤄야 합니다. 고대비 흑백 배경이 효과적입니다.
- 충분히 커야 합니다. Amazon Rekognition Image는 HD 해상도 이미지(1920x1080)에서 40x40픽셀의 작은 얼굴을 감지할 수 있습니다. 이미지 해상도가 높을수록 최소 얼굴 크기를 크게 지정해야 합니다.

- 밝고 선명해야 합니다. [DetectFaces \(p. 243\)](#)를 사용하여 얼굴의 밝기와 선명도를 파악할 수 있습니다.
- 헤드밴드나 마스크 등으로 가리지 않도록 하십시오.

Amazon S3 버킷에 저장된 이미지 분석

Amazon Rekognition Image는 Amazon S3 버킷에 저장된 이미지 또는 이미지 바이트로 제공된 이미지를 분석할 수 있습니다.

이 주제에서는 [DetectLabels \(p. 247\)](#) API 작업을 사용하여 Amazon S3 버킷에 저장된 이미지(JPEG 또는 PNG)에서 객체, 개념, 장면을 감지합니다. [the section called “Image” \(p. 368\)](#) 입력 파라미터를 사용하여 이미지를 Amazon Rekognition Image API 작업에 전달합니다. [Image](#)에서 [S3Object \(p. 387\)](#) 객체 속성을 지정하여 S3 버킷에 저장된 이미지를 참조합니다. Amazon S3 버킷에 저장된 이미지의 이미지 바이트는 base64로 인코딩할 필요가 없습니다. 자세한 내용은 [이미지 \(p. 35\)](#) 단원을 참조하십시오.

S3 객체가 있는 S3 버킷의 리전과 Amazon Rekognition Image 작업에 사용하는 리전이 일치해야 합니다.

[DetectLabels](#)에 대한 이 예제 JSON 요청에서는 MyBucket이라는 Amazon S3 버킷에서 소스 이미지 (`input.jpg`)를 불러옵니다.

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

다음 예제에서는 다양한 AWS SDK와 AWS CLI를 사용하여 [DetectLabels](#)를 호출합니다. [DetectLabels](#) 작업 응답에 대한 자세한 내용은 [DetectLabels 응답 \(p. 101\)](#) 단원을 참조하십시오.

이미지에서 레이블을 감지하려면

1. 아직 설정하지 않았다면 다음과 같이 하십시오.
 - a. [AmazonRekognitionFullAccess](#) 권한과 [AmazonS3ReadOnlyAccess](#) 권한을 가진 IAM 사용자를 만들어가 업데이트합니다. 자세한 내용은 [1단계: AWS 계정 설정 및 IAM 사용자 만들기 \(p. 11\)](#) 단원을 참조하십시오.
 - b. AWS CLI와 AWS SDK를 설치하고 구성합니다. 자세한 내용은 [2단계: AWS CLI 및 AWS SDK 설정 \(p. 11\)](#) 단원을 참조하십시오.
2. 나무, 집, 보트 등과 같은 객체가 한 개 이상 있는 이미지를 S3 버킷에 업로드합니다. 이미지는 `.jpg` 또는 `.png` 형식이어야 합니다.

이에 관한 지침은 Amazon Simple Storage Service 콘솔 사용 설명서의 [Amazon S3에 객체 업로드](#)를 참조하십시오.

3. 다음 예제를 사용하여 [DetectLabels](#) 작업을 호출합니다.

Java

이 예제는 입력 이미지에서 감지된 레이블 목록을 표시합니다. `bucket` 및 `photo`의 값을 2단계에서 사용한 Amazon S3 버킷과 이미지의 이름으로 바꿉니다.

```
//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

이 예제는 detect-labels CLI 작업의 JSON 출력을 표시합니다. bucket 및 photo의 값을 2단계에서 사용한 Amazon S3 버킷과 이미지의 이름으로 바꿉니다.

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

Python

이 예제는 입력 이미지에서 감지된 레이블을 표시합니다. bucket 및 photo의 값을 2단계에서 사용한 Amazon S3 버킷과 이미지의 이름으로 바꿉니다.

```
#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__":
    fileName='input.jpg'
    bucket='bucket'

    client=boto3.client('rekognition')

    response = client.detect_labels(Image={'S3Object':
    {'Bucket':bucket,'Name':fileName}})

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

.NET

이 예제는 입력 이미지에서 감지된 레이블 목록을 표시합니다. bucket 및 photo의 값을 2단계에서 사용한 Amazon S3 버킷과 이미지의 이름으로 바꿉니다.

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

}

로컬 파일 시스템에서 불러온 이미지 분석

Amazon Rekognition Image 작업은 이미지 바이트로 제공된 이미지 또는 Amazon S3 버킷에 저장된 이미지를 분석할 수 있습니다.

이 단원에서는 로컬 파일 시스템에서 로드된 파일을 사용하여 이미지 바이트를 Amazon Rekognition Image API 작업에 제공하는 예제를 설명합니다. [Image \(p. 368\)](#) 입력 파라미터를 사용하여 이미지 바이트를 Rekognition API 작업에 전달합니다. [Image](#)에서 `Bytes` 속성을 지정하여 base64로 인코딩된 이미지 바이트를 전달합니다.

`Bytes` 입력 파라미터를 사용하여 Rekognition API 작업에 전달된 이미지 바이트는 base64로 인코딩되어야 합니다. 이러한 예제의 AWS SDK는 자동으로 base64 인코딩 이미지를 사용합니다. Rekognition API 작업을 호출하기 전에 이미지 바이트를 인코딩할 필요가 없습니다. 자세한 내용은 [이미지 \(p. 35\)](#) 단원을 참조하십시오.

`DetectLabels`에 대한 이 예제 JSON 요청에서 소스 이미지 바이트는 `Bytes` 입력 파라미터로 전달됩니다.

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

다음 예제에서는 다양한 AWS SDK와 AWS CLI를 사용하여 `DetectLabels`를 호출합니다. `DetectLabels` 작업 응답에 대한 자세한 내용은 [DetectLabels 응답 \(p. 101\)](#) 단원을 참조하십시오.

클라이언트 JavaScript 예제는 [JavaScript 사용 \(p. 44\)](#) 단원을 참조하십시오.

로컬 이미지의 레이블을 감지하려면

1. 아직 설정하지 않았다면 다음과 같이 하십시오.
 - a. `AmazonRekognitionFullAccess` 권한과 `AmazonS3ReadOnlyAccess` 권한을 가진 IAM 사용자를 만들어가 업데이트합니다. 자세한 내용은 [1단계: AWS 계정 설정 및 IAM 사용자 만들기 \(p. 11\)](#) 단원을 참조하십시오.
 - b. AWS CLI와 AWS SDK를 설치하고 구성합니다. 자세한 내용은 [2단계: AWS CLI 및 AWS SDK 설정 \(p. 11\)](#) 단원을 참조하십시오.
2. 다음 예제를 사용하여 `DetectLabels` 작업을 호출합니다.

Java

다음 Java 예제에서는 로컬 파일 시스템에서 이미지를 로드하고 `detectLabels` AWS SDK 작업을 사용하여 레이블을 감지하는 방법을 보여줍니다. `photo`의 값을 이미지 파일(.jpg 또는 .png 형식)의 경로와 파일 이름으로 바꿉니다.

```
//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

다음 [Python용 AWS SDK](#) 예제는 로컬 파일 시스템에서 이미지를 로드하고 `detect_labels` 작업을 호출하는 방법을 보여줍니다. `imageFile`의 값을 이미지 파일(.jpg 또는 .png 형식)의 경로와 파일 이름으로 바꿉니다.

```
#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__":
```

```
imageFile='input.jpg'
client=boto3.client('rekognition')

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

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

print('Done...')
```

.NET

다음 예제에서는 로컬 파일 시스템에서 이미지를 로드하고 DetectLabels 작업을 사용하여 레이블을 감지하는 방법을 보여줍니다. photo의 값을 이미지 파일(.jpg 또는 .png 형식)의 경로와 파일 이름으로 바꿉니다.

```
//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

다음 [PHP용 AWS SDK](#) 예제는 로컬 파일 시스템에서 이미지를 로드하고 [DetectFaces API](#) 작업을 호출하는 방법을 보여줍니다. `photo`의 값을 이미지 파일(.jpg 또는 .png 형식)의 경로와 파일 이름으로 바꿉니다.

```
<?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' => $image,
    ),
    'Attributes' => array('ALL')
));
print 'People: Image position and estimated age' . PHP_EOL;
for ($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;  
}  
?>
```

JavaScript 사용

다음 JavaScript 웹 페이지 예제를 이용하여 이미지를 선택하고, 이미지에서 감지된 얼굴의 예상 연령을 확인할 수 있습니다. [the section called “DetectFaces” \(p. 243\)](#)를 호출하면 예상 연령이 반환됩니다.

선택한 이미지는, 이미지를 base64로 인코딩하는 JavaScript `FileReader.readAsDataURL` 함수를 사용하여 로드됩니다. HTML 캔버스에 이미지를 표시할 때 유용한 기능입니다. 그러나 이렇게 하면 Amazon Rekognition Image 작업으로 전달하기 전에 이미지 바이트를 해독해야 합니다. 이 예제에서는 로드된 이미지 바이트를 해독하는 방법을 보여 줍니다. 인코딩된 이미지 바이트가 불편하면 그 대신 로딩된 이미지를 인코딩하지 않는 `FileReader.readAsArrayBuffer`를 사용하십시오. 이렇게 하면 이미지 바이트를 먼저 해독하지 않고도 Amazon Rekognition Image 작업을 호출할 수 있습니다. 문제 해결 예는 [readAsArrayBuffer 사용 \(p. 46\)](#) 단원을 참조하십시오.

JavaScript 예제를 실행하려면

1. 예제 소스 코드를 편집기에 로드합니다.
2. Amazon Cognito 자격 증명 풀 ID를 가져옵니다. 자세한 내용은 [Amazon Cognito 자격 증명 풀 식별자 가져오기 \(p. 47\)](#) 단원을 참조하십시오.
3. 예제 코드의 `AnonLog` 함수에서 `IdentityPoolIdToUse`와 `RegionToUse`를 [Amazon Cognito 자격 증명 풀 식별자 가져오기 \(p. 47\)](#)의 9단계에서 기록해 둔 값으로 바꿉니다.
4. `DetectFaces` 함수에서 `RegionToUse`를 이전 단계에서 사용한 값으로 바꿉니다.
5. 예제 소스 코드를 `.html` 파일로 저장합니다.
6. 그 파일을 브라우저에 로드합니다.
7. 찾아보기... 버튼을 누르고 얼굴이 하나 이상 들어 있는 이미지를 선택합니다. 이미지에서 감지된 각 얼굴의 예상 연령을 담은 표가 나타납니다.

JavaScript 예제 코드

```
<!--  
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>  
    <script src=".app.js"></script>  
    <meta charset="UTF-8">  
    <title>Rekognition</title>  
</head>  
  
<body>  
    <H1>Age Estimator</H1>  
    <input type="file" name="fileToUpload" id="fileToUpload" accept="image/*">  
    <p id="opResult"></p>  
</body>  
<script>  
  
document.getElementById("fileToUpload").addEventListener("change", function (event) {
```

```
    ProcessImage();
}, false);

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

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

readAsArrayBuffer 사용

다음은 샘플 코드에서 ProcessImage 함수를 대신 구현하는 코드 조각입니다. 여기서는 readAsArrayBuffer를 사용하여 이미지를 로드하고 DetectFaces를 호출합니다. readAsArrayBuffer는 로드된 파일을 base64로 인코딩하지 않기 때문에 Amazon Rekognition Image 작업을 호출하기 위해 먼저 이미지 바이트의 인코딩을 해독할 필요가 없습니다.

```
//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>";
                }
            });
        }
    })(theFile);
}
```

```
        document.getElementById("opResult").innerHTML = table;
    }
});

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

Amazon Cognito 자격 증명 풀 식별자 가져오기

간단히 설명하기 위해, 이 예제에서는 익명 Amazon Cognito 자격 증명 풀을 사용하여 Amazon Rekognition Image API에 대한 미인증 액세스 권한을 제공합니다. 이 방법이 현재 상황에 알맞습니다. 예를 들어, 사용자 로그인을 위한 웹 사이트 무료 액세스 권한 또는 평가판 액세스 권한을 미인증 액세스 권한으로 제공할 수 있습니다. 미인증 액세스 권한을 제공하려면 Amazon Cognito 사용자 풀을 사용하십시오. 자세한 내용은 [Amazon Cognito 사용자 풀](#)을 참조하십시오.

다음 절차에서는 인증되지 않은 자격 증명을 사용할 수 있는 자격 증명 풀을 만드는 방법과 예제 코드에 필요한 자격 증명 풀 식별자를 가져오는 방법을 보여 줍니다.

자격 증명 풀 식별자를 가져오려면

1. Amazon Cognito [콘솔](#)을 엽니다.
2. [Create new identity pool]을 선택합니다.
3. 자격 증명 풀 이름*에 자격 증명 풀의 이름을 입력합니다.
4. 인증되지 않은 자격 증명에서 인증되지 않은 자격 증명에 대한 액세스 활성화를 선택합니다.
5. [Create Pool]을 선택합니다.
6. 세부 정보 보기 선택하고 인증되지 않은 자격 증명의 역할 이름을 적어 둡니다.
7. [Allow]를 선택합니다.
8. 플랫폼에서 JavaScript를 선택합니다.
9. AWS 자격 증명 얻기에서 코드 조각에 표시된 자격 증명 풀 ID와 리전을 적어 둡니다.
10. <https://console.aws.amazon.com/iam/>에서 IAM 콘솔을 엽니다.
11. 탐색 창에서 Roles를 선택합니다.
12. 6단계에서 적어 둔 역할 이름을 선택합니다.
13. 권한에서 Attach Policy(정책 연결)를 선택합니다.
14. AmazonRekognitionReadOnlyAccess를 선택합니다.
15. Attach Policy를 선택합니다.

이미지 방향 및 경계 상자 좌표 가져오기

Amazon Rekognition Image를 사용하는 애플리케이션은 일반적으로 Amazon Rekognition Image 작업으로 감지된 이미지와 감지된 얼굴 주위의 상자를 표시해야 합니다. 애플리케이션에서 이미지를 올바르게 표시하려면 이미지의 방향을 확인하고 수정해야 합니다. 일부 .jpg 파일의 경우 이미지의 교환 이미지 파일 형식(Exif) 메타데이터에 이미지의 방향이 포함됩니다. 다른 .jpg 파일 및 모든 .png 파일의 경우 Amazon Rekognition Image 작업은 예상 방향을 반환합니다.

얼굴 주위에 상자를 표시하려면 얼굴의 경계 상자에 대한 좌표가 필요하며 상자의 방향이 올바르지 않으면 해당 좌표를 조정해야 할 수 있습니다. Amazon Rekognition Image 얼굴 감지 작업은 감지된 각 얼굴에 대한 경계 상자 좌표를 반환합니다.

다음 Amazon Rekognition Image 작업은 이미지의 방향 및 경계 상자 좌표를 수정하는데 필요한 정보를 반환합니다.

- [CompareFaces \(p. 219\)](#)

- [DetectFaces \(p. 243\)](#)
- [DetectLabels \(p. 247\)](#)(이미지 방향을 수정하는 데 필요한 정보만 반환)
- [IndexFaces \(p. 287\)](#)
- [RecognizeCelebrities \(p. 304\)](#)

이 예제는 코드에 대한 다음 정보를 가져오는 방법을 보여줍니다.

- 이미지의 예상 방향(Exif 메타데이터에 방향 정보가 없는 경우)
- 이미지에서 감지된 얼굴의 경계 상자 좌표
- 예상 이미지 방향의 영향을 받는 경계 상자의 변환된 경계 상자 좌표

이 예제의 정보를 사용하여 이미지의 방향이 올바른지, 경계 상자가 애플리케이션의 올바른 위치에 표시되는지 확인하십시오.

이미지와 경계 상자를 회전하고 표시하는 데 사용되는 코드는 사용하는 언어와 환경에 따라 다르므로 코드에 이미지와 경계 상자를 표시하는 방법이나 Exif 메타데이터에서 방향 정보를 가져오는 방법을 설명하지 않습니다.

이미지의 방향 찾기

애플리케이션에서 이미지를 올바르게 표시하려면 이미지를 회전해야 할 수 있습니다. 다음 이미지의 방향은 0도이며 올바르게 표시되어 있습니다.



그러나 다음 이미지는 시계 반대 방향으로 90도 회전된 상태입니다. 올바르게 표시하려면 이미지의 방향을 찾고 코드에서 해당 정보를 사용하여 이미지를 0도로 회전해야 합니다.



.jpg 형식의 일부 이미지는 Exif 메타데이터에 방향 정보를 포함합니다. 작업의 응답에서 OrientationCorrection 필드의 값이 null인 경우 이미지의 Exif 메타데이터에 방향이 포함됩니다. Exif 메타데이터의 orientation 필드에서 이미지의 방향을 찾을 수 있습니다. Amazon Rekognition Image는 Exif 메타데이터에서 이미지 방향 정보의 유무를 식별하지만 해당 정보에 대한 액세스를 제공하지 않습니다. 이미지의 Exif 메타데이터에 액세스하려면 타사 라이브러리를 사용하거나 고유한 코드를 작성하십시오. 자세한 내용은 [Exif 버전 2.31](#) 단원을 참조하십시오.

.png 형식의 이미지에는 Exif 메타데이터가 없습니다. Exif 메타데이터가 없는 .jpg 이미지와 모든 .png 이미지의 경우 Amazon Rekognition Image 작업은 OrientationCorrection 필드에 이미지의 예상 방향을 반환합니다. 예상 방향은 시계 반대 방향으로 90도 씩 측정됩니다. 예를 들어 Amazon Rekognition Image는 방향이 0도인 이미지의 경우 ROTATE_0을, 시계 반대 방향으로 90도 회전한 이미지의 경우 ROTATE_90을 반환합니다.

Note

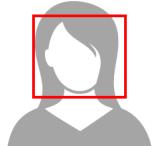
CompareFaces 작업은 SourceImageOrientationCorrection 필드에 원본 이미지 방향을, TargetImageOrientationCorrection 필드에 대상 이미지 방향을 반환합니다.

이미지의 방향을 알면 이미지를 회전하고 올바르게 표시하도록 코드를 작성할 수 있습니다.

경계 상자 표시

이미지 속의 얼굴을 분석하는 Amazon Rekognition Image 작업은 얼굴을 둘러싸는 경계 상자의 좌표도 반환합니다. 자세한 내용은 [Bounding Box \(p. 344\)](#) 단원을 참조하십시오.

애플리케이션의 다음 이미지에 표시된 상자와 비슷한 경계 상자를 얼굴 주위에 표시하려면 코드에서 경계 상자 좌표를 사용하십시오. 작업에서 반환된 경계 상자 좌표는 이미지의 방향을 반영합니다. 이미지를 올바르게 표시하기 위해 회전해야 하는 경우 경계 상자 좌표를 변환해야 할 수 있습니다.



Exif 메타데이터에 방향 정보가 없는 경우 경계 상자 표시

이미지에 Exif 메타데이터가 없거나 Exif 메타데이터의 `orientation` 필드가 채워져 있지 않은 경우 Amazon Rekognition Image 작업은 다음을 반환합니다.

- 이미지의 예상 방향
- 예상 방향으로 향한 경계 상자 좌표

이미지를 올바르게 표시하기 위해 회전해야 하는 경우 경계 상자도 회전해야 합니다.

예를 들어 다음 이미지의 방향은 시계 반대 방향으로 90도이며 얼굴 주위에 경계 상자를 표시합니다. 경계 상자는 Amazon Rekognition Image 작업에서 반환된 예상 방향에 대한 좌표를 사용하여 표시됩니다.



이미지를 0도 방향으로 회전할 때는 경계 상자 좌표를 변환하여 경계 상자도 회전해야 합니다. 예를 들어 다음 이미지는 시계 반대 방향으로 90도에서 0도로 회전되었습니다. 경계 상자 좌표가 아직 변환되지 않았기 때문에 경계 상자는 잘못된 위치에 표시되어 있습니다.



Exif 메타데이터에 방향이 없는 경우 경계 상자를 회전하여 표시하려면

1. 최소 한 명의 얼굴은 있으나 Exif 메타데이터 방향이 없는 입력 이미지를 제공하는 Amazon Rekognition Image 작업을 호출하십시오. 관련 예제는 [이미지에서 얼굴 감지 \(p. 106\)](#) 단원을 참조하십시오.
2. 응답의 `OrientationCorrection` 필드에 반환된 예상 방향을 확인하십시오.
3. 코드의 2단계에서 적어 둔 예상 방향을 사용하여 이미지를 0도 방향으로 회전하십시오.
4. 위쪽 및 왼쪽 경계 상자 좌표를 0도 방향으로 변환하고 코드에서 이미지의 픽셀 점으로 변환하십시오. 다음 목록에서 2단계에서 적어 둔 예상 방향과 일치하는 수식을 사용하십시오.

다음 정의를 참조하십시오.

- `ROTATE_(n)`은 Amazon Rekognition Image 작업에서 반환된 예상 이미지 방향입니다.
- `<face>`는 Amazon Rekognition Image 작업에서 반환된 얼굴에 대한 정보를 나타냅니다. 예를 들어 [DetectFaces \(p. 243\)](#) 작업에서 반환한 `FaceDetail (p. 359)` 데이터 유형에는 원본 이미지에서 감지된 얼굴에 대한 경계 상자 정보가 포함되어 있습니다.
- `image.width` 및 `image.height`는 원본 이미지의 너비와 높이에 대한 픽셀 값입니다.

- 경계 상자 좌표는 이미지 크기에 따라 0과 1 사이의 값입니다. 예를 들어 방향이 0도인 이미지의 경우 BoundingBox.left 값을 0.9로 설정하면 왼쪽 좌표가 이미지의 오른쪽에 가깝게 됩니다. 상자를 표시하려면 경계 상자 좌표 값을 이미지의 픽셀 점으로 변환하고 다음 각 수식에 표시된 대로 0도로 회전하십시오. 자세한 내용은 [BoundingBox \(p. 344\)](#) 단원을 참조하십시오.

ROTATE_0

```
left = image.width*BoundingBox.Left
```

```
top = image.height*BoundingBox.Top
```

ROTATE_90

```
left = image.height * (1 - (<face>.BoundingBox.Top +  
<face>.BoundingBox.Height))
```

```
top = image.width * <face>.BoundingBox.Left
```

ROTATE_180

```
left = image.width - (image.width*(<face>.BoundingBox.Left  
+<face>.BoundingBox.Width))
```

```
top = image.height * (1 - (<face>.BoundingBox.Top +  
<face>.BoundingBox.Height))
```

ROTATE_270

```
left = image.height * BoundingBox.top
```

```
top = image.width * (1 - BoundingBox.Left - BoundingBox.Width)
```

- 다음 수식을 사용하여 코드에서 경계 상자의 너비와 높이를 이미지의 픽셀 범위로 계산하십시오.

경계 상자의 너비와 높이는 BoundingBox.Width 및 BoundingBox.Height 필드에 반환됩니다. 너비 및 높이 값의 범위는 이미지 크기에 따라 0과 1 사이입니다. image.width 및 image.height는 원본 이미지의 너비와 높이에 대한 픽셀 값입니다.

```
box width = image.width * (<face>.BoundingBox.Width)
```

```
box height = image.height * (<face>.BoundingBox.Height)
```

- 4단계와 5단계에서 계산한 값을 사용하여 회전된 이미지에 경계 상자를 표시하십시오.

Exif 메타데이터에 방향 정보가 있는 경우 경계 상자 표시

이미지의 방향이 Exif 메타데이터에 포함되어 있는 경우 Amazon Rekognition Image 작업은 다음을 수행합니다.

- 작업 응답의 방향 수정 필드에 null을 반환하십시오. 이미지를 회전하려면 코드에서 Exif 메타데이터에 제공된 방향을 사용하십시오.
- 이미 0도로 향한 경계 상자 좌표를 반환하십시오. 경계 상자를 옮바른 위치에 표시하려면 반환된 좌표를 사용하십시오. 해당 좌표를 변환할 필요가 없습니다.

예제: 이미지 방향 및 이미지의 경계 상자 좌표 가져오기

다음 예제에서는 Java용 AWS SDK를 사용하여 이미지의 예상 방향을 확인하고 DetectFaces 작업에서 감지된 얼굴의 경계 상자 좌표를 변환하는 방법을 보여줍니다.

이 예제는 로컬 파일 시스템에서 이미지를 로드하고 DetectFaces 작업을 호출하고 이미지의 높이와 너비를 결정하며 회전된 이미지에 대한 얼굴의 경계 상자 좌표를 계산합니다. 이 예제에서는 Exif 메타데이터에 저장된 방향 정보를 처리하는 방법을 보여주지 않습니다.

이 코드를 사용하려면 photo 값을 로컬에 저장된 이미지(.png 또는 .jpg 형식)의 이름과 경로로 바꾸십시오.

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.)  
  
package aws.example.rekognition.image;  
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.util.IOUtils;  
import com.amazonaws.services.rekognition.model.AgeRange;  
import com.amazonaws.services.rekognition.model.AmazonRekognitionException;  
import com.amazonaws.services.rekognition.model.Attribute;  
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;  
  
public class RotateImage {  
  
    public static void main(String[] args) throws Exception {  
  
        String photo = "input.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 detect faces and show face age and placement

try{
    DetectFacesRequest request = new DetectFacesRequest()
        .withImage(new Image()
            .withBytes((imageBytes)))
        .withAttributes(Attribute.ALL);

    DetectFacesResult result = amazonRekognition.detectFaces(request);
    System.out.println("Orientation: " + result.getOrientationCorrection() + "\n");
    List <FaceDetail> faceDetails = result.getFaceDetails();

    for (FaceDetail face: faceDetails) {
        System.out.println("Face:");
        ShowBoundingBoxPositions(height,
            width,
            face.getBoundingBox(),
            result.getOrientationCorrection());
        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();
    }
} 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

이 예제에서는 PIL/Pillow 이미지 라이브러리를 사용하여 이미지 너비와 높이를 확인합니다. 자세한 내용은 [Pillow](#)를 참조하십시오. 이 예제는 애플리케이션에서 필요할 수 있는 exif 메타데이터를 유지합니다. exif 메타데이터를 저장하지 않도록 선택하면 호출에서 DetectFaces로 예상 방향이 반환됩니다.

```
#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 ShowBoundingBoxPositions(imageHeight, imageWidth, box, rotation):
    left = 0
    top = 0

    if rotation == 'ROTATE_0':
        left = imageWidth * box['Left']
        top = imageHeight * box['Top']

    if rotation == 'ROTATE_90':
        left = imageHeight * (1 - (box['Top'] + box['Height']))
        top = imageWidth * box['Left']

    if rotation == 'ROTATE_180':
        left = imageWidth - (imageWidth * (box['Left'] + box['Width']))
        top = imageHeight * (1 - (box['Top'] + box['Height']))

    if rotation == 'ROTATE_270':
        left = imageHeight * box['Top']
        top = imageWidth * (1 - box['Left'] - box['Width'])
```

```
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']))

if __name__ == "__main__":
    photo='input.png'
    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.detect_faces(Image={'Bytes': image_binary}, Attributes=['ALL'])

    print('Detected faces for ' + photo)
    for faceDetail in response['FaceDetails']:
        print ('Face:')
        if 'OrientationCorrection' in response:
            print('Orientation: ' + response['OrientationCorrection'])
            ShowBoundingBoxPositions(height, width, faceDetail['BoundingBox'],
            response['OrientationCorrection'])

        else:
            print ('No estimated orientation. Check Exif data')

        print('The detected face is estimated to be between ' +
        str(faceDetail['AgeRange']['Low'])
        + ' and ' + str(faceDetail['AgeRange']['High']) + ' years')
    print()
```

.NET

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

public class ImageOrientationAndBoundingBox
{
    public static void Example()
    {
        String photo = "photo.jpg";

        AmazonRekognitionClient rekognitionClient = new AmazonRekognitionClient();

        Image image = new 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;
        }

        int height;
        int width;
        // Used to extract original photo width/height
        using (System.Drawing.Bitmap imageBitmap = new System.Drawing.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
        {
            DetectFacesRequest detectFacesRequest = new DetectFacesRequest()
            {
                Image = image,
                Attributes = new List<String>() { "ALL" }
            };

            DetectFacesResponse detectFacesResponse =
rekognitionClient.DetectFaces(detectFacesRequest);
            foreach (FaceDetail face in detectFacesResponse.FaceDetails)
            {
                Console.WriteLine("Face:");
                ShowBoundingBoxPositions(height, width,
                    face.BoundingBox, detectFacesResponse.OrientationCorrection);
                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("The detected face is estimated to be between " +
                    face.AgeRange.Low + " and " + face.AgeRange.High + " years old.");
                Console.WriteLine();
            }
        }
    }
}
```

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

public static void ShowBoundingBoxPositions(int imageHeight, int imageWidth,
    BoundingBox box, String rotation)
{
    float left = 0;
    float top = 0;

    if (rotation == null)
    {
        Console.WriteLine("No estimated 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);
}

}
```

저장된 비디오 작업

Amazon Rekognition Video는 비디오를 분석할 때 사용할 수 있는 API입니다. Amazon Rekognition Video를 사용하여 Amazon Simple Storage Service (Amazon S3) 버킷에 저장된 비디오에서 레이블, 얼굴, 사람, 유명 인사 및 성인(선정적이고 노골적인) 콘텐츠를 찾아낼 수 있습니다. 미디어/엔터테인먼트 및 공공 안전 같은 범주에서 Amazon Rekognition Video를 사용할 수 있습니다. 전에는 객체나 사람의 비디오를 스캔하는 데 여러 시간이 걸렸으며, 사람이 눈으로 보는 데 따른 실수도 있었습니다. Amazon Rekognition Video는 항목과 비디오 전체에서 그 시점을 자동으로 알아냅니다.

이 단원에서는 Amazon Rekognition Video가 수행할 수 있는 유형의 감지과 인식, API 개요, Amazon Rekognition Video 사용 예제를 다룹니다.

항목

- [감지 및 인식 유형 \(p. 57\)](#)
- [Amazon Rekognition Video API 개요 \(p. 57\)](#)
- [Amazon Rekognition Video 작업 불러오기 \(p. 59\)](#)
- [Amazon Rekognition Video 구성 \(p. 64\)](#)
- [Java 또는 Python으로 Amazon S3 버킷에 저장된 비디오 분석\(SDK\) \(p. 66\)](#)
- [AWS Command Line Interface으로 비디오 분석 \(p. 72\)](#)
- [자습서: Amazon Rekognition Lambda 함수 생성 \(p. 75\)](#)
- [참조: 비디오 분석 결과 알림 \(p. 81\)](#)
- [Amazon Rekognition Video 문제 해결 \(p. 82\)](#)

감지 및 인식 유형

Amazon Rekognition Video를 사용하여 비디오에서 다음 정보를 분석할 수 있습니다.

- [레이블 \(p. 100\)](#)
- [얼굴 \(p. 105\)](#)
- [사람 \(p. 167\)](#)
- [유명 인사 \(p. 173\)](#)
- [선정적이고 노골적인 성인 콘텐츠 \(p. 187\)](#)

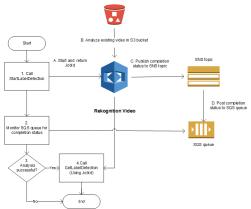
자세한 내용은 [Amazon Rekognition: 작동 방식 \(p. 4\)](#) 단원을 참조하십시오.

Amazon Rekognition Video API 개요

Amazon Rekognition Video는 Amazon S3 버킷에 저장된 비디오를 처리합니다. 디자인 패턴은 비동기 작업 세트입니다. 먼저 [StartLabelDetection \(p. 330\)](#) 같은 start 작업을 불러와 비디오 분석을 시작합니다. 요청 완료 상태가 Amazon Simple Notification Service (Amazon SNS) 주제에 게시됩니다. Amazon SNS 주제에서 완료 상태를 가져오려면 Amazon Simple Queue Service (Amazon SQS) 대기열 또는 AWS Lambda 기

능을 사용하면 됩니다. 완료 상태가 되면 [GetLabelDetection \(p. 278\)](#) 같은 Get 작업을 불러와 요청 결과를 가져옵니다.

다음 그림은 Amazon S3 버킷에 저장된 비디오에서 레이블을 감지하는 프로세스를 보여줍니다. 그림에서 Amazon SQS 대기열은 Amazon SNS 주제의 완료 상태를 가져옵니다. 또는 AWS Lambda 기능을 사용할 수도 있습니다.



이 프로세스는 얼굴과 사람을 찾는 경우와 동일합니다. 다음 표에는 비-스토리지 Amazon Rekognition 작업의 Start 및 Get 작업이 나열되어 있습니다.

감지	시작 작업	가져오기 작업
사람	StartPersonTracking (p. 334)	GetPersonTracking (p. 282)
얼굴	StartFaceDetection (p. 322)	GetFaceDetection (p. 268)
레이블	StartLabelDetection (p. 330)	GetLabelDetection (p. 278)
유명 인사	StartCelebrityRecognition (p. 315)	GetCelebrityRecognition (p. 259)
노골적이거나 선정적인 성인 콘텐츠	StartContentModeration (p. 318)	GetContentModeration (p. 264)

[GetCelebrityRecognition](#) 이외의 Get 작업의 경우 입력 비디오 전체에서 엔터티가 감지되는 시점의 추적 정보를 Amazon Rekognition Video가 반환합니다.

Amazon Rekognition Video 사용에 대한 자세한 내용은 [Amazon Rekognition Video 작업 불러오기 \(p. 59\)](#) 단원을 참조하십시오. Amazon SQS를 사용하여 비디오를 분석하는 예제는 [Java](#) 또는 [Python](#)으로 Amazon S3 버킷에 저장된 비디오 분석(SDK) (p. 66) 단원을 참조하십시오. AWS CLI 예제는 [AWS Command Line Interface](#)로 비디오 분석 (p. 72) 단원을 참조하십시오.

비디오 형식과 스토리지

Amazon Rekognition 작업으로 Amazon S3 버킷에 저장된 비디오를 분석할 수 있습니다. 비디오는 H.264 코덱을 사용하여 인코딩해야 합니다. 지원되는 파일 형식은 MPEG-4와 MOV입니다.

코덱은 빠른 전달을 위해 데이터를 압축하고 수신한 데이터를 원래 형태로 압축을 해제하는 소프트웨어 또는 하드웨어입니다. H.264 코덱은 비디오 콘텐츠의 녹화, 압축 및 배포에 흔히 사용됩니다. 비디오 파일 형식에는 하나 이상의 코덱이 포함될 수 있습니다. MOV 또는 MPEG-4 형식의 비디오 파일이 Amazon Rekognition Video로 작동되지 않을 경우에는 비디오 인코딩에 사용된 코덱이 H.264인지 확인하십시오.

저장된 비디오의 최대 파일 크기는 8GB입니다.

사람 검색

컬렉션에 저장된 얼굴 메타데이터를 사용하여 비디오에서 사람을 검색할 수 있습니다. 예를 들어, 보관된 감시 비디오에서 특정 사람이나 여러 사람을 검색할 수 있습니다. [IndexFaces \(p. 287\)](#) 작업을 사용하여 컬렉션에 원본 이미지의 얼굴 메타데이터를 저장합니다. 그런 다음 [StartFaceSearch \(p. 326\)](#)를 사용하여 컬렉션에 원본 이미지의 얼굴 메타데이터를 저장합니다. 그런 다음 [StartFaceSearch \(p. 326\)](#)를 사용하여 컬렉션에 원본 이미지의 얼굴 메타데이터를 저장합니다.

션에서 비동기 얼굴 검색을 시작할 수 있습니다. [GetFaceSearch \(p. 273\)](#)를 사용하여 검색 결과를 가져옵니다. 자세한 내용은 [저장된 비디오에서 얼굴 검색 \(p. 160\)](#) 단원을 참조하십시오. 사람 검색은 스토리지 기반 Amazon Rekognition 작업의 예입니다. 자세한 내용은 [스토리지 기반 API 작업 \(p. 8\)](#) 단원을 참조하십시오.

스트리밍 비디오에서도 사람을 검색할 수 있습니다. 자세한 내용은 [스트리밍 비디오 작업 \(p. 84\)](#) 단원을 참조하십시오.

Amazon Rekognition Video 작업 불러오기

Amazon Rekognition Video는 Amazon Simple Storage Service (Amazon S3) 버킷에 저장된 비디오를 분석할 때 사용할 수 있는 비동기 API입니다. 먼저 [StartPersonTracking \(p. 334\)](#) 같은 Amazon Rekognition Video Start 작업을 불러와 비디오 분석을 시작합니다. Amazon Rekognition Video는 Amazon Simple Notification Service (Amazon SNS) 주제에 분석 요청 결과를 게시합니다. Amazon Simple Queue Service (Amazon SQS) 대기열 또는 AWS Lambda 기능을 사용하여 Amazon SNS 주제에서 비디오 분석의 완료 상태를 가져올 수 있습니다. 끝으로 [GetPersonTracking \(p. 282\)](#) 같은 Amazon Rekognition Get 작업을 불러와 비디오 분석 요청 결과를 가져옵니다.

이후 단원에서는 레이블 감지 작업을 사용하여 Amazon S3 버킷에 저장된 비디오에서 Amazon Rekognition Video가 레이블(객체, 이벤트, 개념 및 활동)을 감지하는 방법을 설명합니다. 다른 Amazon Rekognition Video 작업(—예: [StartFaceDetection \(p. 322\)](#) 및 [StartPersonTracking \(p. 334\)](#))도 동일하게 적용됩니다. 예제 [Java 또는 Python으로 Amazon S3 버킷에 저장된 비디오 분석\(SDK\) \(p. 66\)](#)은 Amazon SQS 대기열을 사용하여 Amazon SNS 주제에서 완료 상태를 가져와 비디오를 분석하는 방법을 보여줍니다. 이것은 [인물 추적 \(p. 167\)](#) 같은 다른 Amazon Rekognition Video 예제의 기준으로도 사용됩니다. AWS CLI 예제는 [AWS Command Line Interface으로 비디오 분석 \(p. 72\)](#) 단원을 참조하십시오.

항목

- [비디오 분석 시작 \(p. 59\)](#)
- [Amazon Rekognition Video 분석 요청의 완료 상태 가져오기 \(p. 60\)](#)
- [Amazon Rekognition Video 분석 결과 가져오기 \(p. 61\)](#)

비디오 분석 시작

[StartLabelDetection \(p. 330\)](#)을 불러와 Amazon Rekognition Video 레이블 감지 요청을 시작합니다. 다음은 [StartLabelDetection](#)으로 전달되는 JSON 요청의 예입니다.

```
{  
    "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"  
}
```

입력 파라미터 Video에는 비디오 파일 이름과 거기서 검색할 Amazon S3 버킷이 있습니다. NotificationChannel에는 비디오 분석 요청이 끝날 때 Amazon Rekognition Video가 알려주는

Amazon SNS 주제의 Amazon Resource Name(ARN)이 있습니다. Amazon SNS 주제는 블러울 Amazon Rekognition Video 앤드포인트와 동일한 AWS 리전에 있어야 합니다. `NotificationChannel`에는 Amazon Rekognition Video가 Amazon SNS 주제를 게시할 때 사용할 수 있는 역할의 ARN도 있습니다. IAM 서비스 역할을 생성하여 귀하의 Amazon SNS 주제에 Amazon Rekognition 게시 권한을 줍니다. 자세한 내용은 [Amazon Rekognition Video 구성 \(p. 64\)](#) 단원을 참조하십시오.

또는 Amazon SNS 주제에 게시된 완료 상태에서 작업을 식별할 때 사용할 수 있는 선택사항인 입력 파라미터 `JobTag`도 지정할 수 있습니다.

분석 작업이 실수로 중복되지 않도록 `idempotent` 토큰 `ClientRequestToken`을 선택적으로 제공할 수 있습니다. `ClientRequestToken` 값을 제공하면 `Start` 작업에서 `StartLabelDetection` 같은 시작 작업의 여러 동일한 호출에 대해 동일한 `JobId`를 반환합니다. `ClientRequestToken` 토큰은 수명이 7일입니다. 7일 후에 다시 사용할 수 있습니다. 토큰 수명 기간 동안 토큰을 재사용할 경우 다음과 같은 현상이 생깁니다.

- 동일한 `Start` 작업과 동일한 입력 파라미터로 토큰을 다시 사용할 경우 동일한 `JobId`가 반환됩니다. 이 작업은 다시 실행되지 않으며 Amazon Rekognition Video는 등록된 Amazon SNS 주제로 완료 상태를 전송하지 않습니다.
- 동일한 `Start` 작업에서 약간의 입력 파라미터를 변경하여 토큰을 재사용할 경우 `idempotentparameterismatchexception`(HTTP 상태 코드: 400) 예외가 표시됩니다.
- 다른 `Start` 작업에 토큰을 재사용할 경우 작업이 진행됩니다.

`StartLabelDetection` 작업에 대한 응답은 작업 식별자입니다(`JobId`). Amazon Rekognition Video가 Amazon SNS 주제에 완료 상태를 게시한 후에 `JobId`를 사용하여 요청을 추적하고 분석 결과를 가져옵니다. 예:

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

너무 많은 작업을 동시에 시작하면서 `StartLabelDetection`을 호출할 경우에는 동시 실행 작업의 수가 Amazon Rekognition 서비스 제한 미만이 될 때까지 `LimitExceededException` 예외(HTTP 상태 코드: 400)가 발생합니다.

다수의 작업으로 인해 `LimitExceededException` 예외가 발생하는 경우에는 Amazon SQS 대기열을 사용하여 수신되는 요청을 관리하는 것이 좋습니다. Amazon SQS 대기열로도 평균 동시 요청 수를 관리하지 못하여 계속해서 `LimitExceededException` 예외가 수신되면 AWS Support에 문의하십시오.

Amazon Rekognition Video 분석 요청의 완료 상태 가져오기

Amazon Rekognition Video는 등록된 Amazon SNS 주제로 분석 완료 알림을 보냅니다. 알림에는 JSON 문자열 형태의 작업 완료 상태와 작업 식별자가 포함됩니다. 성공적인 비디오 분석 요청은 `SUCCEEDED` 상태를 갖습니다. 예를 들어, 다음 결과는 레이블 감지 작업의 성공적인 처리를 보여줍니다.

```
{  
    "JobId": "270c1cc5e1d0ea2fbc59d97cb69a72a5495da75851976b14a1nnnnnnnnnnnn",  
    "Status": "SUCCEEDED",  
    "API": "StartLabelDetection",  
    "JobTag": "DetectingLabels",  
    "Timestamp": 1510865364756,  
    "Video": {  
        "S3ObjectName": "video.mp4",  
        "S3Bucket": "bucket"  
    }  
}
```

자세한 내용은 [참조: 비디오 분석 결과 알림 \(p. 81\)](#) 단원을 참조하십시오.

Amazon Rekognition Video가 Amazon SNS 주제에 게시한 상태 정보를 가져오려면 다음 옵션 중 하나를 사용합니다.

- AWS Lambda – Amazon SNS 주제에 기록하는 AWS Lambda 기능을 신청할 수 있습니다. Amazon Rekognition이 Amazon SNS 주제에 요청 완료 사실을 알릴 때 이 기능이 호출됩니다. 서버 측 코드로 비디오 분석 요청 결과를 처리해야 할 경우에 Lambda 기능을 사용합니다. 예를 들어, 서버 측 코드를 사용하여 비디오에 주석을 달거나 비디오 콘텐츠에 관한 보고서를 생성한 후에 정보를 클라이언트 애플리케이션으로 반환해야 하는 경우가 있을 수 있습니다. Amazon Rekognition API가 대용량 데이터를 반환할 수 있으므로 대용량 비디오는 서버 측 처리를 권장합니다.
- Amazon Simple Queue Service – Amazon SQS 대기열을 Amazon SNS 주제에 신청할 수 있습니다. 그런 다음 Amazon SQS 대기열을 폴링하여 비디오 분석 요청이 완료될 때 Amazon Rekognition이 게시하는 완료 상태를 검색합니다. 자세한 내용은 [Java 또는 Python으로 Amazon S3 버킷에 저장된 비디오 분석 \(SDK\) \(p. 66\)](#) 단원을 참조하십시오. Amazon Rekognition Video 작업을 클라이언트 애플리케이션에서만 불러와야 할 경우에는 Amazon SQS 대기열을 사용합니다.

Important

Amazon Rekognition Video Get 작업을 반복해서 불러와 요청 완료 상태를 가져오는 것은 좋지 않습니다. 그 이유는 요청이 너무 많을 경우 Amazon Rekognition Video가 Get 작업을 제한하기 때문입니다. 여러 비디오를 동시에 처리할 경우 각 비디오의 상태를 개별적으로 확인하기 위해 Amazon Rekognition Video를 폴링하는 것 보다는 하나의 SQS 대기열을 보고 완료 알림을 모니터링하는 것 이 더 효율적이고 간단합니다.

Amazon Rekognition Video 분석 결과 가져오기

비디오 분석 요청의 결과를 가져오려면 먼저 Amazon SNS 주제에서 검색된 완료 상태가 SUCCEEDED인지 확인합니다. 그런 다음 GetLabelDetection을 호출하면 이에 의해 StartLabelDetection에서 반환된 jobId 값이 통과됩니다. 요청 JSON은 다음 예제와 비슷합니다.

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

JobId는 비디오 분석 작업용 ID입니다. 비디오 분석 시 대용량 데이터가 생성될 수 있으므로 MaxResults를 사용하여 단일 Get 작업에서 반환할 결과의 최대수를 지정합니다. MaxResults의 기본값은 1000입니다. 1,000보다 큰 값을 지정한 경우 최대 1,000개의 결과가 반환됩니다. 작업에서 전체 결과 세트가 반환되지 않을 경우에는 그 다음 페이지의 페이지 토큰이 작업 응답으로 반환됩니다. 이전의 Get 요청에서 페이지 토큰을 받은 경우에는 이것을 NextToken과 함께 사용하여 결과의 다음 페이지를 가져옵니다.

Note

Amazon Rekognition은 비디오 분석 작업의 결과를 7일 동안 보유합니다. 이 후에는 분석 결과를 검색하지 못합니다.

GetLabelDetection 작업 응답 JSON은 다음과 비슷합니다.

```
{  
    "JobStatus": "SUCCEEDED",  
    "Labels": [  
        {  
            "Label": {  
                "Confidence": 56.49449920654297,  
                "Name": "Bowl"  
            }  
        }  
    ]  
}
```

```
        },
        "Timestamp": 0
    },
    {
        "Label": {
            "Confidence": 77.5353012084961,
            "Name": "Clothing"
        },
        "Timestamp": 0
    },
    {
        "Label": {
            "Confidence": 53.91270065307617,
            "Name": "Guitar"
        },
        "Timestamp": 0
    },
    {
        "Label": {
            "Confidence": 53.91270065307617,
            "Name": "Musical Instrument"
        },
        "Timestamp": 0
    },
    {
        "Label": {
            "Confidence": 77.5353012084961,
            "Name": "Overcoat"
        },
        "Timestamp": 0
    },
    {
        "Label": {
            "Confidence": 50.50969696044922,
            "Name": "Person"
        },
        "Timestamp": 0
    },
    {
        "Label": {
            "Confidence": 67.22470092773438,
            "Name": "Pumpkin"
        },
        "Timestamp": 0
    },
    {
        "Label": {
            "Confidence": 99.03849792480469,
            "Name": "Speech"
        },
        "Timestamp": 0
    },
    {
        "Label": {
            "Confidence": 67.22470092773438,
            "Name": "Squash"
        },
        "Timestamp": 0
    },
    {
        "Label": {
            "Confidence": 77.5353012084961,
            "Name": "Suit"
        },
        "Timestamp": 0
    }
}
```

```
],
"NextToken": "WlKfzzED1PzBLyAmTpda655kICVnPPAtQ/8V9McI6097JKjqP08uQun6j6fwGEaJFDUQVawYsg==",
"VideoMetadata": {
    "Codec": "h264",
    "DurationMillis": 67301,
    "FileExtension": "mp4",
    "Format": "QuickTime / MOV",
    "FrameHeight": 1080,
    "FrameRate": 29.970029830932617,
    "FrameWidth": 1920
}
}
```

결과를 감지 시간별로(비디오 시작 후 경과한 밀리초) 또는 감지 엔터티(객체, 얼굴, 유명 인사, 종재 레이블 또는 사람)의 알파벳 순으로 정렬할 수 있습니다. 시간별로 정렬하려면 sortBy 입력 파라미터 값을 TIMESTAMP로 설정합니다. SortBy가 지정되지 않으면 기본적으로 시간별로 정렬됩니다. 앞의 예제는 시간별로 정렬된 경우입니다. 엔터티별로 정렬하려면 SortBy 입력 파라미터를 수행할 작업에 해당하는 값과 함께 사용합니다. 예를 들어, GetLabelDetection 호출에서 감지된 레이블별로 정렬하려면 NAME 값을 사용합니다. 다음 예제는 감지된 레이블 Apparel(옷)과 badge(배지)의 레이블 이름별 정렬을 보여줍니다.

```
{
    "JobStatus": "SUCCEEDED",
    "Labels": [
        {
            "Label": {
                "Confidence": 50.53730010986328,
                "Name": "Apparel"
            },
            "Timestamp": 46813
        },
        {
            "Label": {
                "Confidence": 50.538700103759766,
                "Name": "Apparel"
            },
            "Timestamp": 47013
        },
        {
            "Label": {
                "Confidence": 50.76940155029297,
                "Name": "Apparel"
            },
            "Timestamp": 47213
        },
        {
            "Label": {
                "Confidence": 50.504798889160156,
                "Name": "Apparel"
            },
            "Timestamp": 63229
        },
        {
            "Label": {
                "Confidence": 54.01129913330078,
                "Name": "Badge"
            },
            "Timestamp": 8775
        },
        {
            "Label": {
                "Confidence": 55.24839782714844,
                "Name": "Badge"
            },
        }
    ]
}
```

```
        "Timestamp": 8975
    },
    {
        "Label": {
            "Confidence": 60.499000549316406,
            "Name": "Badge"
        },
        "Timestamp": 9175
    },
    {
        "Label": {
            "Confidence": 60.73529815673828,
            "Name": "Badge"
        },
        "Timestamp": 9376
    },
    {
        "Label": {
            "Confidence": 60.914302825927734,
            "Name": "Badge"
        },
        "Timestamp": 9576
    },
    {
        "Label": {
            "Confidence": 50.74089813232422,
            "Name": "Badge"
        },
        "Timestamp": 9776
    }
],
"NextToken": "CmUTsWamUKmnzPvN+FB0REjh+xv6+iY3H0IXrHCn3yMSW1zuOcWyLKob2JjvfPjznalo3m/MGw==",
"VideoMetadata": {
    "Codec": "h264",
    "DurationMillis": 67301,
    "FileExtension": "mp4",
    "Format": "QuickTime / MOV",
    "FrameHeight": 1080,
    "FrameRate": 29.970029830932617,
    "FrameWidth": 1920
}
}
```

Amazon Rekognition Video 구성

비디오가 저장된 Amazon Rekognition Video API를 사용하려면 Amazon SNS 주제에 액세스하도록 IAM 사용자 및 IAM 서비스를 구성해야 합니다. 다음 절차에서는 이 단원의 예시에 사용할 Amazon Rekognition Video를 구성하는 방법을 보여줍니다. 이를 위해 다음을 수행합니다.

- IAM 사용자를 생성하거나 IAM 사용자에게 Amazon Rekognition Video API 액세스를 부여합니다. 이번 예에서는 전체 Amazon Rekognition API에 대한 모든 액세스가 부여되지만 필요한 경우 액세스를 제한할 수 있습니다.
- 사용하려는 AWS SDK를 설치하고 구성합니다.
- IAM 서비스 역할을 생성하여 Amazon Rekognition Video가 분석 요청 완료 상태를 Amazon SNS 주제에 게시하도록 허용합니다.
- Amazon SNS 메시지를 검색할 Amazon SQS 대기열에 예시 코드 액세스를 부여합니다. 예시 코드가 대기열의 메시지를 삭제하기 때문에 전체 액세스가 필요합니다.
- 저장된 비디오 파일을 포함하는 Amazon S3 버킷에 예시 코드 읽기 액세스를 부여합니다.

Note

이 단원의 예시에서는 여러 개의 주제에 Amazon Rekognition Video 액세스를 부여하는 지침을 사용하여 새로운 Amazon SNS 주제를 생성합니다. 기존 Amazon SNS 주제를 사용하고자 하는 경우 3단계의 [기존 Amazon SNS 주제에 대한 액세스 권한 부여 \(p. 66\)](#)을 사용합니다.

Amazon Rekognition Video를 구성하려면

1. Amazon Rekognition Video에 액세스하도록 AWS 계정을 설정합니다. 자세한 내용은 [1단계: AWS 계정 설정 및 IAM 사용자 만들기 \(p. 10\)](#) 단원을 참조하십시오.

사용자에게 최소한 다음 권한이 있는지 확인합니다.

- AmazonSQSFullAccess
- AmazonRekognitionFullAccess
- AmazonS3ReadOnlyAccess

2. 필요한 AWS SDK를 설치하고 구성합니다. 자세한 내용은 [2단계: AWS CLI 및 AWS SDK 설정 \(p. 11\)](#) 단원을 참조하십시오.
3. IAM 서비스 역할을 생성하여 Amazon SNS 주제에 대한 Amazon Rekognition Video 액세스를 부여합니다. 서비스 역할의 Amazon Resource Name(ARN)을 적어둡니다. 자세한 내용은 [여러 Amazon SNS 주제에 대한 액세스 권한 부여 \(p. 65\)](#) 단원을 참조하십시오.

4. 1단계에서 생성한 IAM 사용자에게 [다음 인라인 정책을 추가합니다.](#)

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Sid": "MySid",  
            "Effect": "Allow",  
            "Action": "iam:PassRole",  
            "Resource": "arn:Service role ARN from step 3"  
        }  
    ]  
}
```

인라인 정책에 이름을 지정합니다.

5. 이제 [Java 또는 Python으로 Amazon S3 버킷에 저장된 비디오 분석\(SDK\) \(p. 66\)](#) 및 [AWS Command Line Interface으로 비디오 분석 \(p. 72\)](#)에서 예시를 실행할 수 있습니다.

여러 Amazon SNS 주제에 대한 액세스 권한 부여

IAM 서비스 역할을 사용하여 Amazon Rekognition Video에게 자신이 생성한 Amazon SNS 주제에 대한 액세스 권한을 부여합니다. IAM은 Amazon Rekognition Video 서비스 역할 생성에 필요한 Rekognition 사용 사례를 제공합니다.

`AmazonRekognitionServiceRole` 권한 정책을 사용하여, 주제 이름에 `AmazonRekognition`을 추가하여 (—예: `AmazonRekognitionMyTopicName`) Amazon Rekognition Video에게 여러 Amazon SNS 주제에 대한 액세스 권한을 부여할 수 있습니다.

Amazon Rekognition Video에게 여러 Amazon SNS 주제에 대한 액세스 권한을 부여하려면

1. [IAM 서비스 역할을 만듭니다.](#) 다음 정보를 사용하여 IAM 역할을 생성합니다.

1. 서비스 이름으로 [Rekognition]을 선택합니다.

2. 서비스 역할 사용 사례로 [Rekognition]을 선택합니다. 나열된 AmazonRekognitionServiceRole 권한 정책을 확인합니다. AmazonRekognitionServiceRole은 Amazon Rekognition Video가 AmazonRekognition으로 시작하는 Amazon SNS 주제에 대해 액세스할 수 있도록 허용합니다.
3. 서비스 역할에 이름을 지정합니다.
2. 서비스 역할의 ARN을 기록합니다. 이것은 비디오 분석 작업을 시작할 때 필요합니다.

기존 Amazon SNS 주제에 대한 액세스 권한 부여

Amazon Rekognition Video이 기존 Amazon SNS 주제에 액세스할 수 있도록 허용하는 권한 정책을 만들 수 있습니다.

Amazon Rekognition Video에 기존 Amazon SNS 주제에 대한 액세스 권한을 부여하려면

1. [IAM JSON 정책 편집기로 새 권한 정책을 만들고 다음 정책을 사용합니다.](#) `topicarn`을 원하는 Amazon SNS 주제의 Amazon Resource Name(ARN)으로 바꿉니다.

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

2. [IAM 서비스 역할을 만들거나 기존 IAM 서비스 역할을 업데이트합니다.](#) 다음 정보를 사용하여 IAM 역할을 생성합니다.
 1. 서비스 이름으로 [Rekognition]을 선택합니다.
 2. 서비스 역할 사용 사례로 [Rekognition]을 선택합니다.
 3. 1단계에 만든 권한 정책을 연결합니다.
 3. 서비스 역할의 ARN을 기록합니다. 이것은 비디오 분석 작업을 시작할 때 필요합니다.

Java 또는 Python으로 Amazon S3 버킷에 저장된 비디오 분석(SDK)

이 절차는 Amazon Rekognition Video 레이블 감지 작업, Amazon S3 버킷에 저장된 비디오, Amazon SNS 주제를 사용하여 비디오에서 레이블을 감지하는 방법을 보여줍니다. 또한 이 절차는 Amazon SQS 대기열을 사용하여 Amazon SNS 주제에서 완료 상태를 가져오는 방법도 보여줍니다. 자세한 내용은 [Amazon Rekognition Video 작업 불러오기 \(p. 59\)](#) 단원을 참조하십시오. Amazon SQS 대기열 사용에는 제한이 없습니다. 예를 들어, AWS Lambda 기능을 사용하여 완료 상태를 가져올 수 있습니다. 자세한 내용은 [Amazon SNS 알림을 사용하여 Lambda 함수 호출](#) 단원을 참조하십시오.

이 절차는 [Amazon SNS 콘솔](#)을 사용하여 다음을 실행하는 방법을 보여줍니다.

- Amazon SNS 주제 생성.
- Amazon SQS 대기열 생성
- Amazon Rekognition Video에게 비디오 분석 작업의 완료 상태를 Amazon SNS 주제에 게시할 권한 부여
- Amazon SQS 대기열을 Amazon SNS 주제로 신청

Note

이 절차는 모든 비디오 분석 요청에 대해 단일 Amazon SQS 대기열 및 단일 Amazon SNS 주제를 사용합니다.

이 절차의 예제 코드는 다음을 실행하는 방법을 보여줍니다.

1. [StartLabelDetection \(p. 330\)](#)을 불러와 비디오 분석 요청을 시작합니다.
2. Amazon SQS 대기열에서 완료 상태를 가져옵니다. 이 예제는 [StartLabelDetection](#)에서 반환되는 작업 식별자(Job ID)를 추적하여 완료 상태에서 판독되는 작업 식별자와 일치하는 결과만 가져옵니다. 이 점은 다른 애플리케이션에서 동일한 대기열과 주제를 사용할 경우에 중요하게 고려해야 합니다. 간소화를 위해 이 예제에서는 일치하지 않는 작업을 삭제합니다. 그러한 작업은 Amazon SQS 배달 못한 편지 대기 열에 추가하여 나중에 검사해 보십시오.
3. [GetLabelDetection \(p. 278\)](#)을 불러와 비디오 분석 결과를 가져오고 표시합니다.

사전 조건

이 절차를 실행하려면 AWS SDK for Java을 설치해야 합니다. 자세한 내용은 [Amazon Rekognition 시작하기 \(p. 10\)](#) 단원을 참조하십시오. 사용하는 AWS 계정에 Amazon Rekognition API에 대한 액세스 권한이 있어야 합니다. 자세한 내용은 [Amazon Rekognition API 권한: 작업, 권한 및 리소스 참조 \(p. 31\)](#) 단원을 참조하십시오.

비디오에서 레이블을 감지하려면

1. Amazon Rekognition Video에 대한 사용자 액세스를 구성하고 Amazon SNS에 대한 Amazon Rekognition Video 액세스를 구성합니다. 자세한 내용은 [Amazon Rekognition Video 구성 \(p. 64\)](#) 단원을 참조하십시오.
2. [Amazon SNS 콘솔](#)을 사용하여 [Amazon SNS 주제를 만듭니다](#). 주제 이름에 AmazonRekognition을 추가합니다. 주제 Amazon Resource Name(ARN)을 기록합니다. 사용하는 AWS 엔드포인트와 동일한 리전에 주제가 있어야 합니다.
3. [Amazon SQS 콘솔](#)을 사용하여 [Amazon SQS 표준 대기열을 만듭니다](#). 대기열 ARN을 기록합니다.
4. 2단계에서 만든 주제에 대기열을 신청합니다.
5. [Amazon SQS 대기열에 메시지를 전송하도록 Amazon SNS 주제에 권한 부여](#).
6. .mp4, .mov 또는 .avi 형식 비디오 파일을 S3 버킷으로 업로드합니다. 테스트할 때는 30초 이하의 비디오를 업로드합니다.
7. 다음 AWS SDK for Java 코드를 사용하여 비디오에서 레이블을 감지합니다.
 - topicArn, roleArn 및 queueUrl을 앞에서 기록했던 Amazon SNS 주제 ARN, IAM 역할 ARN 및 Amazon SQS 대기열 URL로 바꿉니다.
 - 값 bucket 및 video를 6단계에서 지정한 버킷과 비디오 파일 이름으로 바꿉니다.

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.)

package com.amazonaws.samples;
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.LabelDetection;
import com.amazonaws.services.rekognition.model.LabelDetectionSortBy;
import com.amazonaws.services.rekognition.model.NotificationChannel;
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.sqs.AmazonSQS;
import com.amazonaws.services.sqs.AmazonSQSClientBuilder;
import com.amazonaws.services.sqs.model.Message;
import com.fasterxml.jackson.databind.JsonNode;
import com.fasterxml.jackson.databind.ObjectMapper;
import java.util.*;

public class VideoDetect {

    private static String bucket = "";
    private static String video = "";
    private static String queueUrl = "";
    private static String topicArn="";
    private static String roleArn="";
    private static AmazonSQS sqs = null;
    private static AmazonRekognition rek = null;

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

    private static String startJobId = null;

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

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

    //=====
    StartLabels(bucket, video);
    //=====
    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(queueUrl).getMessages();
        if (dotLine++<20){
            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")){
                        //=====
                        GetResultsLabels();
                        //=====
                    }
                    else{
                        System.out.println("Video analysis failed");
                    }

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

                else{
                    System.out.println("Job received was not job " +
startJobId);
                    //Delete unknown message. Consider moving message to dead
letter queue
                    sqs.deleteMessage(queueUrl,message.getReceiptHandle());
                }
            }
        }
    }
```

```
        } while (!jobFound);

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

private static void StartLabels(String bucket, String video) throws Exception{
    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 GetResultsLabels() 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();
            System.out.print("Millisecond: " + Long.toString(seconds) + " ");
            System.out.println("\t" + detectedLabel.getLabel().getName() +
"\t" +
detectedLabel.getLabel().getConfidence().toString());
            System.out.println();
        }
    }
}
```

```
        }
    } while (labelDetectionResult != null &&
labelDetectionResult.getNextToken() != null);

}
}
```

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.)

import boto3
import json
import sys

class VideoDetect:
    jobId = ''
    rek = boto3.client('rekognition')
    queueUrl = ''
    roleArn = ''
    topicArn = ''
    bucket = ''
    video = ''

    def main(self):

        jobFound = False
        sqs = boto3.client('sqS')

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

                                                NotificationChannel={'RoleArn':
self.roleArn, 'SNSTopicArn': self.topicArn})
        #=====
        print('Start Job Id: ' + response['JobId'])
        dotLine=0
        while jobFound == False:
            sqsResponse = sqs.receive_message(QueueUrl=self.queueUrl,
MessageAttributeNames=['ALL'],
                                                MaxNumberOfMessages=10)

            if sqsResponse:

                if 'Messages' not in sqsResponse:
                    if dotLine<20:
                        print('.', end='')
                        dotLine=dotLine+1
                    else:
                        print()
                        dotLine=0
                    sys.stdout.flush()
                    continue

                for message in sqsResponse['Messages']:
                    notification = json.loads(message['Body'])
                    rekMessage = json.loads(notification['Message'])
                    print(rekMessage['JobId'])
                    print(rekMessage['Status'])
                    if str(rekMessage['JobId']) == response['JobId']:
```

```
        print('Matching Job Found: ' + rekMessage['JobId'])
        jobFound = True
        =====
        self.GetResultsLabels(rekMessage['JobId'])
        =====

        sqs.delete_message(QueueUrl=self.queueUrl,
                           ReceiptHandle=message['ReceiptHandle'])

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

        print('done')

    def GetResultsLabels(self, jobId):
        maxResults = 10
        paginationToken = ''
        finished = False

        while finished == False:
            response = self.rek.get_label_detection(JobId=jobId,
                                                     MaxResults=maxResults,
                                                     NextToken=paginationToken,
                                                     SortBy='TIMESTAMP')

            print(response['VideoMetadata']['Codec'])
            print(str(response['VideoMetadata']['DurationMillis']))
            print(response['VideoMetadata']['Format'])
            print(response['VideoMetadata']['FrameRate'])

            for labelDetection in response['Labels']:
                print(labelDetection['Label']['Name'])
                print(labelDetection['Label']['Confidence'])
                print(str(labelDetection['Timestamp']))

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

    if __name__ == "__main__":
        analyzer=VideoDetect()
        analyzer.main()
```

- 코드를 작성하고 실행합니다. 이 작업은 마치는 데 시간이 걸릴 수 있습니다. 작업이 끝나면 비디오에서 감지된 레이블 목록이 표시됩니다. 자세한 내용은 [비디오에서 레이블 감지 \(p. 102\)](#) 단원을 참조하십시오.

AWS Command Line Interface으로 비디오 분석

AWS Command Line Interface (AWS CLI)을 사용하여 Amazon Rekognition Video 작업을 불러올 수 있습니다. 디자인 패턴은 Amazon Rekognition Video API를 AWS SDK for Java 또는 다른 AWS SDK와 함께 사용

할 때와 동일합니다. 자세한 내용은 [Amazon Rekognition Video API 개요 \(p. 57\)](#) 단원을 참조하십시오. 다음 절차는 AWS CLI를 사용하여 비디오에서 레이블을 감지하는 방법을 보여줍니다.

`start-label-detection`을 불러와 비디오에서 레이블 감지를 시작합니다. Amazon Rekognition이 비디오 분석을 마치면 `start-label-detection`의 `--notification-channel` 파라미터에 지정된 Amazon SNS 주제로 그 완료 상태가 전송됩니다. Amazon Simple Queue Service (Amazon SQS) 대기열을 Amazon SNS 주제로 신청하여 완료 상태를 가져올 수 있습니다. 그런 다음 `receive-message`를 폴링하여 Amazon SQS 대기열에서 완료 상태를 가져옵니다.

완료 상태 알림은 `receive-message` 응답 내의 JSON 구조입니다. 응답에서 JSON을 추출해야 합니다. 완료 상태 JSON에 관한 내용은 [참조: 비디오 분석 결과 알림 \(p. 81\)](#) 단원을 참조하십시오. 완료 상태 JSON의 `Status` 필드 값이 `SUCCEEDED`이면 `get-label-detection`을 불러와 비디오 분석 요청의 결과를 가져올 수 있습니다.

다음 절차에는 Amazon SQS 대기열을 폴링할 코드가 포함되지 않습니다. 그리고 Amazon SQS 대기열에서 반환되는 JSON의 구문 분석용 코드도 포함되지 않습니다. Java 예제는 [Java 또는 Python으로 Amazon S3 버킷에 저장된 비디오 분석\(SDK\) \(p. 66\)](#) 단원을 참조하십시오.

사전 조건

이 절차를 실행하려면 AWS CLI를 설치해야 합니다. 자세한 내용은 [Amazon Rekognition 시작하기 \(p. 10\)](#) 단원을 참조하십시오. 사용하는 AWS 계정에 Amazon Rekognition API에 대한 액세스 권한이 있어야 합니다. 자세한 내용은 [Amazon Rekognition API 권한: 작업, 권한 및 리소스 참조 \(p. 31\)](#) 단원을 참조하십시오.

Amazon Rekognition Video를 구성하고 비디오를 업로드하려면

1. Amazon Rekognition Video에 대한 사용자 액세스를 구성하고 Amazon SNS에 대한 Amazon Rekognition Video 액세스를 구성합니다. 자세한 내용은 [Amazon Rekognition Video 구성 \(p. 64\)](#) 단원을 참조하십시오.
2. [Amazon SNS 콘솔](#)을 사용하여 [Amazon SNS 주제를 만듭니다](#). 주제 이름에 AmazonRekognition을 추가합니다. 주제 Amazon Resource Name(ARN)을 기록합니다.
3. [Amazon SQS 콘솔](#)을 사용하여 [Amazon SQS 표준 대기열을 만듭니다](#). 대기열 ARN을 기록합니다.
4. 2단계에서 만든 주제에 대기열을 신청합니다.
5. [Amazon SQS 대기열에 메시지를 전송하도록 Amazon SNS 주제에 권한 부여](#).
6. .mp4, .mov 또는 .avi 형식 비디오 파일을 S3 버킷으로 업로드합니다. 개발 및 테스트를 진행할 때는 30초 이하의 짧은 비디오를 사용하는 것이 좋습니다.

이에 관한 지침은 Amazon Simple Storage Service 콘솔 사용 설명서의 [Amazon S3에 객체 업로드](#)를 참조하십시오.

비디오에서 레이블을 감지하려면

1. 다음 AWS CLI 명령을 실행하여 비디오에서 레이블 감지를 시작합니다.

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

다음 값을 업데이트합니다.

- `bucketname` 및 `videofile`을 6단계에서 지정한 Amazon S3 버킷 이름과 파일 이름으로 바꿉니다.
- `Endpoint` 및 `us-east-1`을 사용하는 AWS 엔드포인트와 리전으로 변경합니다.
- `TopicARN`을 이전 절차 2단계에서 만든 Amazon SNS 주제의 ARN으로 변경합니다.

- RoleARN를 이전 절차 1단계에서 만든 IAM 역할의 ARN으로 변경합니다.
 - RekognitionUser를 Amazon Rekognition Video 작업을 불러올 권한이 있는 AWS 계정으로 변경합니다.
2. 응답의 JobId 값을 기록합니다. 이 응답은 다음 JSON 예제와 비슷해 보입니다.

```
{  
    "JobId": "547089ce5b9a8a0e7831afa655f42e5d7b5c838553f1a584bf350ennnnnnnnnn"  
}
```

3. 완료 상태 JSON의 Amazon SQS 대기열을 폴링할 코드를 적습니다([receive-message](#)를 사용하여).
4. 코드를 적어 완료 상태 JSON에서 Status 필드를 추출합니다.
5. Status 값이 SUCCESS이면 다음 AWS CLI 명령을 실행하여 레이블 감지 결과를 표시합니다.

```
aws rekognition get-label-detection --job-id JobId \  
--endpoint-url Endpoint \  
--region us-east-1 \  
--profile RekognitionUser
```

다음 값을 업데이트합니다.

- JobId를 변경하여 2단계에서 기록한 작업 식별자와 일치시킵니다.
- Endpoint 및 us-east-1을 사용하는 AWS 엔드포인트와 리전으로 변경합니다.
- RekognitionUser를 Amazon Rekognition Video 작업을 불러올 권한이 있는 AWS 계정으로 변경합니다.

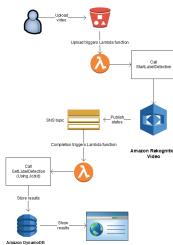
그 결과는 다음 예제 JSON과 비슷해 보입니다.

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

자습서: Amazon Rekognition Lambda 함수 생성

이 자습서에서는 Java Lambda 함수를 사용하여 레이블 감지용 비디오 분석 작업의 결과를 가져오는 방법을 보여 줍니다.

Amazon Rekognition Video 작업에 Lambda 함수를 사용할 수 있습니다. 예를 들어 다음 디어그램은 Amazon S3 버킷에 업로드되면 Lambda 함수를 사용하여 비디오 분석을 자동으로 시작하는 웹 사이트를 보여 줍니다. Lambda 함수가 트리거되면 [the section called "StartLabelDetection" \(p. 330\)](#)을 호출하여 업로드된 비디오에서 레이블 감지를 시작합니다. 두 번째 Lambda 함수는 분석 완료 상태가 등록된 Amazon SNS 주제에 전송되며 트리거됩니다. 두 번째 Lambda 함수는 [the section called "GetLabelDetection" \(p. 278\)](#)을 호출하여 분석 결과를 가져옵니다. 그러면 결과가 데이터베이스에 저장되어 웹 사이트에 표시될 준비가 됩니다.



이 자습서에서는 Amazon Rekognition Video가 등록된 Amazon SNS 주제로 비디오 분석의 완료 상태를 보내면 Lambda 함수가 트리거됩니다. 그런 다음 [the section called "GetLabelDetection" \(p. 278\)](#)을 호출하여 비디오 분석 결과를 수집합니다. 설명을 위해 이 자습서에서는 레이블 감지 결과를 CloudWatch 로그에 쓰습니다. 애플리케이션의 Lambda 함수에서 나중에 사용할 수 있도록 분석 결과를 저장해야 합니다. 예를 들어 Amazon DynamoDB를 사용하여 분석 결과를 저장할 수 있습니다. 자세한 내용은 [DynamoDB 사용](#)을 참조하십시오.

다음 절차에서는 이 방법을 보여 줍니다.

- Amazon SNS 주제를 만들고 권한을 설정합니다.
- AWS Management 콘솔을 사용하여 Lambda 함수를 만들고 Amazon SNS 주제를 구독합니다.
- AWS Management 콘솔을 사용하여 Lambda 함수를 구성합니다.
- AWS Toolkit for Eclipse 프로젝트에 샘플 코드를 추가하고 이를 Lambda 함수에 업로드합니다.
- AWS CLI를 사용하여 Lambda 함수를 테스트합니다.

사전 조건

이 자습서에서는 사용자가 AWS Toolkit for Eclipse를 잘 알고 있는 것으로 가정합니다. 자세한 내용은 [AWS Toolkit for Eclipse](#)를 참조하십시오.

SNS 주제 생성

Amazon Rekognition Video 비디오 분석 작업의 완료 상태는 Amazon SNS 주제에 전송됩니다. 이 절차에서 Amazon SNS 주제와, Amazon Rekognition Video에 Amazon SNS 주제에 대한 액세스 권한을 부여하는 IAM 서비스 역할을 생성합니다. 자세한 내용은 [Amazon Rekognition Video 작업 불러오기 \(p. 59\)](#) 단원을 참조하십시오.

Amazon SNS 주제를 생성하려면

- 아직 생성하지 않았다면 Amazon Rekognition Video에게 자신의 Amazon SNS 주제에 대한 액세스 권한을 부여할 IAM 서비스 역할을 생성합니다. Amazon 리소스 이름(ARN)을 적어둡니다. 자세한 내용은 [여러 Amazon SNS 주제에 대한 액세스 권한 부여 \(p. 65\)](#) 단원을 참조하십시오.

- Amazon SNS 콘솔을 사용하여 [Amazon SNS 주제를 만듭니다](#). 주제 이름에 AmazonRekognition을 추가합니다. 주제 ARN을 기록합니다.

Lambda 함수 생성

AWS Management 콘솔을 사용하여 Lambda 함수를 생성합니다. 그런 다음 AWS Toolkit for Eclipse 프로젝트를 사용하여 AWS Lambda에 Lambda 함수 패키지를 업로드합니다. AWS Toolkit for Eclipse에 Lambda 함수를 생성할 수도 있습니다. 자세한 내용은 [자습서: AWS Lambda 함수 생성, 업로드 및 호출 방법](#)을 참조하십시오.

Lambda 함수 생성 방법

- AWS Management Console에 로그인한 후 <https://console.aws.amazon.com/lambda/>에서 AWS Lambda 콘솔을 열니다.
- [Create function]을 선택합니다.
- [Author from scratch]를 선택합니다.
- 이름*에 함수의 이름을 입력합니다.
- 실행 시간*에서 Java 8을 선택합니다.
- 역할*에서 사용자 지정 역할 생성을 선택합니다. 사용자 지정 역할에 대한 새 탭이 표시됩니다.
- 새 탭에서 다음을 수행합니다.
 - IAM 역할을 선택한 후 새 IAM 역할 생성을 선택합니다.
 - 역할 이름에 새 사용자 지정 역할의 이름을 입력합니다.
 - 허용을 선택하여 새 역할을 만듭니다. 사용자 지정 역할 탭이 닫히고 Lambda 생성 페이지로 돌아옵니다.
- [Role*]에서 [Choose an existing role]을 선택합니다.
- 기존 역할*에서 7단계에서 만든 역할을 선택합니다.
- [Create function]을 선택합니다.

Lambda 함수 구성

Lambda 함수를 생성한 후 [SNS 주제 생성 \(p. 75\)](#)에서 생성하는 Amazon SNS 주제에 의해 트리거되도록 구성합니다. Lambda 함수의 제한 시간 및 메모리 요구 사항도 조정합니다.

Lambda 함수를 구성하려면

- 함수 코드에서 핸들러에 com.amazonaws.lambda.demo.JobCompletionHandler를 입력합니다.
- 기본 설정의 메모리에서 1024를 선택합니다.
- 기본 설정의 제한 시간에서 10초를 선택합니다.
- 디자이너의 트리거 추가에서 SNS를 선택합니다.
- 트리거 구성에서 [SNS 주제 생성 \(p. 75\)](#)에서 생성한 Amazon SNS 주제를 선택합니다.
- 트리거 활성화를 선택합니다.
- 트리거를 추가하려면 추가를 선택합니다.
- [Save]를 선택합니다.

IAM Lambda 규칙 구성

Amazon Rekognition Video 작업을 호출하려면 IAM Lambda 역할에 AmazonRekognitionFullAccess AWS 관리형 정책을 추가합니다. [the section called "StartLabelDetection" \(p. 330\)](#) 등의 Start 작업에도 Amazon

Rekognition Video가 Amazon SNS 주제에 액세스하는 데 사용하는 IAM 서비스 역할에 대한 역할 전달 권한이 필요합니다.

역할을 구성하려면

1. AWS Management 콘솔에 로그인한 다음 <https://console.aws.amazon.com/iam/>에서 IAM 콘솔을 엽니다.
2. 탐색 창에서 Roles를 선택합니다.
3. 목록에서 [Lambda 함수 생성 \(p. 76\)](#)에서 생성한 사용자 지정 역할의 이름을 선택합니다.
4. Permissions 탭을 선택합니다.
5. Attach policy(정책 연결)를 선택합니다.
6. 정책 목록에서 AmazonRekognitionFullAccess를 선택합니다.
7. Attach policy(정책 연결)를 선택합니다.
8. 다시 사용자 지정 역할을 선택합니다.
9. 페이지의 하단으로 스크롤하고 인라인 정책 추가를 선택합니다.
10. [JSON] 탭을 선택합니다.
11. 기존 정책을 다음 정책으로 바꿉니다. [servicerole](#)을 [SNS 주제 생성 \(p. 75\)](#)에서 생성한 IAM 서비스 역할로 바꿉니다.

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Sid": "mysid",  
            "Effect": "Allow",  
            "Action": "iam:PassRole",  
            "Resource": "arn:aws:iam::123456789012:role/servicerole"  
        }  
    ]  
}
```

12. [Review policy]를 선택합니다.
13. 이름*에 정책 이름을 입력합니다.
14. [Create policy]를 선택합니다.

AWS Toolkit for Eclipse Lambda 프로젝트 만들기

Lambda 함수가 트리거되면 다음 코드가 Amazon SNS 주제에서 완료 상태를 가져오고 [the section called "GetLabelDetection" \(p. 278\)](#)을 호출하여 분석 결과를 가져옵니다. 감지된 레이블 수와 감지된 레이블 목록이 CloudWatch 로그에 기록됩니다. Lambda 함수는 나중에 사용할 수 있도록 비디오 분석 결과를 저장해야 합니다.

AWS Toolkit for Eclipse Lambda 프로젝트를 만들려면

1. [AWS Toolkit for Eclipse AWS Lambda 프로젝트를 만듭니다.](#)
 - 프로젝트 이름:에 선택한 프로젝트 이름을 입력합니다.
 - 입력 유형:에서 SNS 이벤트를 선택합니다.
 - 다른 필드는 바꾸지 않고 그대로 둡니다.
2. 이클립스 프로젝트 탐색기에서 생성된 Lambda 핸들러 메서드를 열고 내용을 다음으로 바꿉니다.

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

```
package com.amazonaws.lambda.demo;

import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.LambdaLogger;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.SNSEvent;
import java.util.List;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.rekognition.AmazonRekognition;
import com.amazonaws.services.rekognition.AmazonRekognitionClientBuilder;
import com.amazonaws.services.rekognition.model.GetLabelDetectionRequest;
import com.amazonaws.services.rekognition.model.GetLabelDetectionResult;
import com.amazonaws.services.rekognition.model.LabelDetection;
import com.amazonaws.services.rekognition.model.LabelDetectionSortBy;
import com.amazonaws.services.rekognition.model.VideoMetadata;
import com.fasterxml.jackson.databind.JsonNode;
import com.fasterxml.jackson.databind.ObjectMapper;

public class JobCompletionHandler implements RequestHandler<SNSEvent, String> {

    @Override
    public String handleRequest(SNSEvent event, Context context) {

        String message = event.getRecords().get(0).getSNS().getMessage();
        LambdaLogger logger = context.getLogger();

        // Parse SNS event for analysis results. Log results
        try {
            ObjectMapper operationResultMapper = new ObjectMapper();
            JsonNode jsonResultTree = operationResultMapper.readTree(message);
            logger.log("Rekognition Video Operation:=====");
            logger.log("Job id: " + jsonResultTree.get("JobId"));
            logger.log("Status : " + jsonResultTree.get("Status"));
            logger.log("Job tag : " + jsonResultTree.get("JobTag"));
            logger.log("Operation : " + jsonResultTree.get("API"));

            if (jsonResultTree.get("API").asText().equals("StartLabelDetection")) {

                if (jsonResultTree.get("Status").asText().equals("SUCCEEDED")){
                    GetResultsLabels(jsonResultTree.get("JobId").asText(), context);
                }
                else{
                    String errorMessage = "Video analysis failed for job "
                        + jsonResultTree.get("JobId")
                        + "State " + jsonResultTree.get("Status");
                    throw new Exception(errorMessage);
                }
            }
            else
                logger.log("Operation not StartLabelDetection");

        } catch (Exception e) {
            logger.log("Error: " + e.getMessage());
            throw new RuntimeException (e);

        }
        return message;
    }

    void GetResultsLabels(String startJobId, Context context) throws Exception {
}
```

```
LambdaLogger logger = context.getLogger();

AmazonRekognition rek =
AmazonRekognitionClientBuilder.standard().withRegion(Regions.US_EAST_1).build();

int maxResults = 1000;
String paginationToken = null;
GetLabelDetectionResult labelDetectionResult = null;
String labels = "";
Integer labelsCount = 0;
String label = "";
String currentLabel = "";

//Get label detection results and log them.
do {

    GetLabelDetectionRequest labelDetectionRequest = new
GetLabelDetectionRequest().withJobId(startJobId)

    .withSortBy(LabelDetectionSortBy.NAME).withMaxResults(maxResults).withNextToken(paginationToken);

    labelDetectionResult = rek.getLabelDetection(labelDetectionRequest);

    paginationToken = labelDetectionResult.getNextToken();
    VideoMetadata videoMetaData = labelDetectionResult.getVideoMetadata();

    // Add labels to log
    List<LabelDetection> detectedLabels = labelDetectionResult.getLabels();

    for (LabelDetection detectedLabel : detectedLabels) {
        label = detectedLabel.getLabel().getName();
        if (label.equals(currentLabel)) {
            continue;
        }
        labels = labels + label + " / ";
        currentLabel = label;
        labelsCount++;

    }
} while (labelDetectionResult != null && labelDetectionResult.getNextToken() != null);

logger.log("Total number of labels : " + labelsCount);
logger.log("labels : " + labels);

}

}
```

3. Rekognition 네임스페이스가 해결되지 않았습니다. 이를 수정하려면:

- import com.amazonaws.services.rekognition.AmazonRekognition; 행의 밑줄이 표시된 부분 위에서 마우스를 멈춥니다.
 - Fix project set up...(프로젝트 설정 수정...)을 선택합니다.
 - Amazon Rekognition 아카이브의 최신 버전을 선택합니다.
 - 확인을 선택하여 프로젝트에 아카이브를 추가합니다.
4. Eclipse 코드 창에서 마우스 오른쪽 버튼을 클릭하고 [AWS Lambda]와 [Upload function to AWS Lambda]를 차례대로 선택합니다.
5. [Select Target Lambda Function] 페이지에서 사용할 AWS 리전을 선택합니다.

6. Choose an existing lambda function(기존 lambda 함수 선택)을 선택한 후, [Lambda 함수 생성 \(p. 76\)](#)에서 생성한 Lambda 함수를 선택합니다.
7. [Next]를 선택합니다.
8. 함수 구성 페이지에서 [Lambda 함수 생성 \(p. 76\)](#)에서 생성한 IAM 역할을 선택합니다.
9. 완료를 선택하면 Lambda 함수가 AWS에 업로드됩니다.

Lambda 함수 테스트

다음 AWS CLI 명령으로 비디오에 대한 레이블 감지 분석을 시작하여 Lambda 함수를 테스트합니다. 분석을 완료하면 Lambda 함수가 트리거됩니다. CloudWatch Logs 로그를 확인하여 분석이 성공했는지 확인합니다.

Lambda 함수를 테스트하려면

1. .mp4, .mov 또는 .avi 형식 비디오 파일을 S3 버킷으로 업로드합니다. 테스트할 때는 30초 이하의 비디오를 업로드합니다.

이에 관한 지침은 Amazon Simple Storage Service 콘솔 사용 설명서의 [Amazon S3에 객체 업로드](#)를 참조하십시오.

2. 다음 AWS CLI 명령을 실행하여 비디오에서 레이블 감지를 시작합니다.

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

다음 값을 업데이트합니다.

- **bucketname** 및 **videofile**을 Amazon S3 버킷 이름과 레이블을 감지할 비디오의 파일 이름으로 변경합니다.
- **Endpoint** 및 **us-east-1**을 사용하는 AWS 엔드포인트와 리전으로 변경합니다.
- **TopicARN**을 [SNS 주제 생성 \(p. 75\)](#)에서 생성한 Amazon SNS 주제의 ARN으로 바꿉니다.
- **RoleARN**을 [SNS 주제 생성 \(p. 75\)](#)에서 생성한 IAM 역할의 ARN으로 바꿉니다.
- **RekognitionUser**를 Amazon Rekognition Video 작업을 불러올 권한이 있는 AWS 계정으로 변경합니다.

3. 응답의 **JobId** 값을 기록합니다. 이 응답은 다음 JSON 예제와 비슷해 보입니다.

```
{  
  "JobId": "547089ce5b9a8a0e7831afa655f42e5d7b5c838553f1a584bf350ennnnnnnnnnnn"  
}
```

4. <https://console.aws.amazon.com/cloudwatch/> 콘솔을 엽니다.
5. 분석이 완료되면 Lambda 함수의 로그 항목이 로그 그룹에 표시됩니다.
6. Lambda 함수를 선택하여 로그 스트림을 봅니다.
7. 최신 로그 스트림을 선택하여 Lambda 함수가 만든 로그 항목을 봅니다. 작업이 성공하면 다음과 같이 나타납니다.

```
Time (UTC +00:00) Message
2018-02-28

19:48:01 START RequestId: 47cb1472-1cc0-11e8-860a-4d0aa2ff96b Version: $LATEST
19:48:02 Rekognition Video Operation: =====
19:48:02 Job id : "9c7c3b1403a375a044c6dbe793d5c78d06014ee16f5efde083ad654b06f6c59a"
19:48:02 Status : "SUCCEEDED"
19:48:02 Job tag : null
19:48:02 Operation : "StartLabelDetection"
19:48:09 Total number of labels : 29
19:48:09 labels : Audience / Badge / Bottle / Clothing / Coat / Crowd / Electric Guitar / Flora / Food / Guitar / Hu
19:48:09 Result : []
19:48:09 END RequestId: 47cb1472-1cc0-11e8-860a-4d0aa2ff96b
19:48:09 REPORT Duration: 17m14.77s 1x0.11e8 860a 4d0aa2ff96b Duration: 8932 ms Billed Duration:
```

작업 id 값은 3단계에서 적어 둔 JobId의 값과 일치해야 합니다.

참조: 비디오 분석 결과 알림

Amazon Rekognition은 완료 상태를 포함해 Amazon Rekognition Video 분석 요청의 결과를 Amazon Simple Notification Service(Amazon SNS) 주제에 게시합니다. Amazon SNS 주제에서 알림을 가져오려면 Amazon Simple Queue Service 대기열 또는 AWS Lambda 기능을 사용합니다. 자세한 정보는 [the section called "Amazon Rekognition Video 작업 불러오기" \(p. 59\)](#) 단원을 참조하십시오. 문제 해결에는 [Java 또는 Python](#)으로 Amazon S3 버킷에 저장된 비디오 분석(SDK) ([p. 66](#)) 단원을 참조하십시오.

페이지로드는 다음 JSON 형식입니다.

```
{
    "JobId": "String",
    "Status": "String",
    "API": "String",
    "JobTag": "String",
    "Timestamp": Number,
    "Video": {
        "S3ObjectName": "String",
        "S3Bucket": "String"
    }
}
```

이름	설명
JobId	작업 식별자입니다. Start 작업에서 반환되는 작업 식별자와 일치합니다(예: StartPersonTracking (p. 334)).
상태	작업의 상태입니다. 유효 값은 SUCCEEDED, FAILED 또는 ERROR입니다.
API	입력 비디오를 분석할 때 사용되는 Amazon Rekognition Video 작업입니다.
JobTag	작업의 식별자입니다. 호출에서 JobTag를 StartLabelDetection (p. 330) 같은 Start 작업으로 지정합니다.
Timestamp	작업이 완료된 시점의 Unix 타임스탬프입니다.
동영상	처리된 비디오에 관한 세부 정보입니다. 파일 이름과 파일이 저장된 Amazon S3 버킷이 포함되어 있습니다.

다음은 Amazon SNS 주제로 전송된 성공적인 알림의 예입니다.

```
{  
    "JobId": "6de014b0-2121-4bf0-9e31-856a18719e22",  
    "Status": "SUCCEEDED",  
    "JobType": "LABEL_DETECTION",  
    "Message": "",  
    "Timestamp": 1502230160926,  
    "Video": {  
        "S3ObjectName": "video.mpg",  
        "S3Bucket": "videobucket"  
    }  
}
```

Amazon Rekognition Video 문제 해결

다음은 Amazon Rekognition Video 및 저장된 비디오 작업 시 문제 해결 정보를 다룹니다.

Amazon SNS 주제로 전송된 완료 상태를 수신할 수 없습니다.

Amazon Rekognition Video는 비디오 분석이 완료되면 상태 정보를 Amazon SNS 주제에 게시합니다. 일반적으로 Amazon SQS 대기열 또는 Lambda 함수를 통해 주제에 구독함으로써 완료 상태 메시지를 받습니다. 조사에 도움이 되도록 이메일로 Amazon SNS 주제를 구독하십시오. 그러면 Amazon SNS 주제로 전송된 메시지를 이메일 수신함에서 받습니다. 자세한 정보는 [주제 구독](#)을 참조하십시오.

애플리케이션에서 메시지를 수신하지 않는 경우 다음을 고려하십시오.

- 분석이 완료되었는지 확인합니다. Get 작업 응답(예: `GetLabelDetection`)의 `JobStatus` 값을 확인합니다. 값이 `IN_PROGRESS`인 경우 분석이 끝난 것이 아닙니다. 따라서 완료 상태도 아직 Amazon SNS 주제에 게시되지 않았습니다.
- Amazon Rekognition Video에 Amazon SNS 주제를 게시할 권한을 부여한 IAM 서비스 역할이 있는지 확인합니다. 자세한 내용은 [Amazon Rekognition Video 구성 \(p. 64\)](#) 단원을 참조하십시오.
- 사용 중인 IAM 서비스 역할이 역할 자격 증명을 사용하여 Amazon SNS 주제에 게시할 수 있는지 확인합니다. 다음 단계를 사용합니다.
- 사용자의 Amazon Resource Name(ARN)을 가져옵니다.

```
aws sts get-caller-identity --profile RekognitionUser
```

- AWS Management Console을 사용하여 사용자 ARN에 역할 신뢰 관계를 추가합니다. 예:

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

- 역할 수임: `aws sts assume-role --role-arn arn:Role ARN --role-session-name SessionName --profile RekognitionUser`

- Amazon SNS 주제에 게시: `aws sns publish --topic-arn arn:Topic ARN --message "Hello World!" --region us-east-1 --profile RekognitionUser`

AWS CLI 명령이 작동하는 경우 메시지를 수신합니다(이메일로 주제를 구독한 경우 이메일 수신함에서 수신). 메시지를 수신하지 못한 경우:

- Amazon Rekognition Video를 구성했는지 확인합니다. 자세한 내용은 [Amazon Rekognition Video 구성 \(p. 64\)](#) 단원을 참조하십시오.
- 이 문제 해결 질문에 대한 기타 팁을 확인합니다.
- 올바른 Amazon SNS 주제를 사용하고 있는지 확인합니다.
 - IAM 서비스 역할을 사용하여 Amazon Rekognition Video에 단일 Amazon SNS 주제에 대한 액세스를 제공한 경우 올바른 Amazon SNS 주제에 대한 권한을 부여했는지 확인합니다. 자세한 내용은 [기존 Amazon SNS 주제에 대한 액세스 권한 부여 \(p. 66\)](#) 단원을 참조하십시오.
 - IAM 서비스 역할을 사용하여 Amazon Rekognition Video에 여러 SNS 주제에 대한 액세스를 제공한 경우 올바른 주제를 사용 중이고, 주제의 이름 앞에 AmazonRekognition이 붙어 있는지 확인합니다. 자세한 내용은 [여러 Amazon SNS 주제에 대한 액세스 권한 부여 \(p. 65\)](#) 단원을 참조하십시오.
- AWS Lambda 함수를 사용하는 경우 Lambda 함수가 올바른 Amazon SNS 주제에 구독되었는지 확인합니다. 자세한 내용은 [Amazon SNS 알림을 사용하여 Lambda 함수 호출](#) 단원을 참조하십시오.
- Amazon SNS 주제에 대해 Amazon SQS 대기열을 구독한 경우 Amazon SNS 주제에 Amazon SQS 대기열로 메시지를 전송할 권한이 있는지 확인합니다. 자세한 내용은 [Amazon SQS 대기열에 메시지를 전송하도록 Amazon SNS 주제에 권한 부여](#) 단원을 참조하십시오.

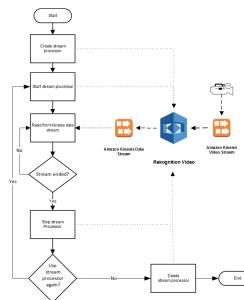
스트리밍 비디오 작업

Amazon Rekognition Video를 사용하여 스트리밍 비디오에서 얼굴을 감지하고 인식할 수 있습니다. 일반적인 사용 사례는 비디오 스트림에서 아는 얼굴을 찾아내려고 할 때입니다. Amazon Rekognition Video는 Amazon Kinesis Video Streams를 사용해 비디오 스트림을 수신하고 처리합니다. Amazon Rekognition Video에서 Kinesis data stream로 분석 결과가 출력되면 클라이언트 애플리케이션에서 읽습니다. Amazon Rekognition Video은 스트리밍 비디오 분석을 시작하고 관리하는 데 사용할 수 있는 스트림 프로세서 ([CreateStreamProcessor \(p. 227\)](#))를 제공합니다.

Note

Amazon Rekognition Video 스트리밍 API는 미국 동부(오하이오) 리전 또는 아시아 태평양(시드니) 리전에서 사용할 수 없습니다.

다음 다이어그램은 Amazon Rekognition Video이 스트리밍 비디오에서 어떻게 얼굴을 감지하고 인식하는지 보여줍니다.



스트리밍 비디오에 Amazon Rekognition Video를 사용하려면 애플리케이션에 다음을 구현해야 합니다.

- Amazon Rekognition Video으로 스트리밍 비디오를 전송하는 Kinesis video stream. 자세한 내용은 [Kinesis video stream](#)을 참조하십시오.
- 스트리밍 비디오 분석을 관리할 Amazon Rekognition Video 스트림 프로세서. 자세한 내용은 [스트리밍 비디오 분석 시작 \(p. 86\)](#) 단원을 참조하십시오.
- Amazon Rekognition Video의 Kinesis data stream로 전송하는 분석 결과를 읽을 Kinesis data stream 소비자. 자세한 내용은 [Amazon Kinesis Streams 소비자](#)를 참조하십시오.

이 단원에서는 Kinesis video stream과 Kinesis data stream을 만들고, Amazon Rekognition Video으로 비디오를 스트리밍하고, 분석 결과를 사용하는 애플리케이션 개발에 대한 정보를 제공합니다. 자세한 내용은 [스트리밍 비디오에서 얼굴 인식 \(p. 84\)](#) 단원을 참조하십시오.

항목

- [스트리밍 비디오에서 얼굴 인식 \(p. 84\)](#)
- [Amazon Rekognition Video에 Kinesis 스트림 액세스 권한 부여 \(p. 85\)](#)
- [스트리밍 비디오 분석 시작 \(p. 86\)](#)
- [스트리밍 비디오 분석 결과 읽기 \(p. 93\)](#)
- [참조: Kinesis 얼굴 인식 레코드 \(p. 95\)](#)

스트리밍 비디오에서 얼굴 인식

Amazon Rekognition Video은 스트리밍 비디오에서 감지한 얼굴과 일치하는 얼굴을 얼굴 모음에서 찾아낼 수 있습니다. 모음에 대한 자세한 내용은 [모음에서 얼굴 검색 \(p. 127\)](#) 단원을 참조하십시오. 다음 절차는 스트리밍 비디오에서 얼굴을 인식하기 위해 거쳐야 하는 단계에 대한 설명입니다.

사전 조건

이 절차를 실행하려면 AWS SDK for Java을 설치해야 합니다. 자세한 내용은 [Amazon Rekognition 시작하기 \(p. 10\)](#) 단원을 참조하십시오. 사용하는 AWS 계정에 Amazon Rekognition API에 대한 액세스 권한이 있어야 합니다. 자세한 내용은 [Amazon Rekognition API 권한: 작업, 권한 및 리소스 참조 \(p. 31\)](#) 단원을 참조하십시오.

비디오 스트림에서 얼굴을 인식하려면(AWS SDK)

1. Amazon Rekognition Video에 Kinesis video stream 및 Kinesis data stream 액세스 권한을 부여하는 IAM 서비스 역할을 아직 안 만들었다면, 지금 만듭니다. ARN을 적어둡니다. 자세한 내용은 [Kinesis Video Streams와 Kinesis Data Streams에 대한 액세스 허용 \(p. 85\)](#) 단원을 참조하십시오.
2. [모음을 만들고 \(p. 128\)](#), 사용한 모음 식별자를 적어둡니다.
3. 2단계에서 만든 모음에 검색하고자 하는 [얼굴을 인덱싱 \(p. 137\)](#)합니다.
4. [Kinesis video stream을 만들고](#) 스트림의 Amazon Resource Name(ARN)을 적어둡니다.
5. [Kinesis data stream을 만듭니다](#). 스트림 이름 앞에 AmazonRekognition을 추가하고 스트림의 ARN을 적어둡니다.
6. [스트림 프로세서를 만듭니다 \(p. 88\)](#). 선택한 이름, Kinesis video stream ARN(4단계), Kinesis data stream ARN(5단계), 모음 식별자(2단계)를 [the section called “CreateStreamProcessor” \(p. 227\)](#)에 파라미터로 전달합니다.
7. 6단계에 선택한 스트림 프로세서 이름을 사용하여 [스트림 프로세서를 시작 \(p. 89\)](#)합니다.
8. [PutMedia](#) 작업을 사용하여 4단계에 만든 Kinesis video stream로 소스 비디오를 스트리밍합니다. 자세한 내용은 [PutMedia API 예제](#)를 참조하십시오.
9. [Amazon Rekognition Video의 분석 출력을 사용합니다 \(p. 93\)](#).

Amazon Rekognition Video에 Kinesis 스트림 액세스 권한 부여

AWS Identity and Access Management(IAM) 서비스 역할을 사용하여 Amazon Rekognition Video에 대한 Kinesis video stream 읽기 액세스와 Kinesis data stream에 대한 쓰기 액세스를 허용합니다.

Kinesis Video Streams와 Kinesis Data Streams에 대한 액세스 허용

IAM은 Rekognition 서비스 역할 사용 사례를 제공합니다. 이 역할을 [AmazonRekognitionServiceRole](#) 권한 정책에 사용할 경우 여러 Kinesis data stream에 쓸 수 있고 모든 Kinesis video stream에서 읽을 수 있습니다. Amazon Rekognition Video에서 여러 Kinesis data stream에 쓰기 액세스할 수 있도록 허용하려면 Kinesis data stream 이름 앞에 AmazonRekognition을 추가하면 됩니다(예: [AmazonRekognitionMyDataStreamName](#))。

Amazon Rekognition Video에 Kinesis video stream 및 Kinesis data stream 액세스 권한을 부여하려면

1. [IAM 서비스 역할을 만듭니다](#). 다음 정보를 사용하여 IAM 역할을 생성합니다.
 1. 서비스 이름으로 [Rekognition]을 선택합니다.
 2. 서비스 역할 사용 사례로 [Rekognition]을 선택합니다.
 3. [AmazonRekognitionServiceRole] 권한 정책을 선택합니다. 이 정책은 Amazon Rekognition Video의 AmazonRekognition으로 시작하는 Kinesis data stream에 대한 쓰기 액세스, 모든 Kinesis video stream에 대한 읽기 액세스를 할 수 있도록 허용합니다.

- 서비스 역할의 Amazon Resource Name(ARN)을 적어둡니다. 이것은 비디오 분석 작업을 시작할 때 필요합니다.

개별 Kinesis 스트림에 대한 액세스 허용

Amazon Rekognition Video이 개별 Kinesis video stream과 Kinesis data stream에 액세스할 수 있도록 허용하는 권한 정책을 만들 수 있습니다.

Amazon Rekognition Video에 개별 Kinesis video stream 및 Kinesis data stream 액세스 권한을 부여하려면

- IAM JSON 정책 편집기로 새 권한 정책을 만들고 다음 정책을 사용합니다. `data-arn`을 원하는 Kinesis data stream의 ARN으로 바꾸고 `video-arn`을 원하는 Kinesis video stream의 ARN으로 바꿉니다.

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

- IAM 서비스 역할을 만들거나 기존 IAM 서비스 역할을 업데이트합니다. 다음 정보를 사용하여 IAM 역할을 생성합니다.
 - 서비스 이름으로 [Rekognition]을 선택합니다.
 - 서비스 역할 사용 사례로 [Rekognition]을 선택합니다.
 - 1단계에 만든 권한 정책을 연결합니다.
 - 서비스 역할의 ARN을 기록합니다. 이것은 비디오 분석 작업을 시작할 때 필요합니다.

스트리밍 비디오 분석 시작

Amazon Rekognition Video 스트림 프로세서를 시작하고 Amazon Rekognition Video로 비디오를 스트리밍하여 스트리밍 비디오 분석을 시작합니다. Amazon Rekognition Video 스트림 프로세서를 사용하면 스트림 프로세서를 시작, 중지하고 관리할 수 있습니다. [CreateStreamProcessor \(p. 227\)](#)를 호출하여 스트림 프로세서를 만들 수 있습니다. 요청 파라미터에는 Kinesis video stream의 Amazon Resource Name(ARN), Kinesis data stream 그리고 스트리밍 비디오에서 얼굴을 인식하는 데 사용되는 모음의 식별자가 포함됩니다. 또한 스트림 프로세서에 지정한 이름도 포함됩니다.

Starts processing a stream processor. You create a stream processor by calling `CreateStreamProcessor` (p. 227). To tell `StartStreamProcessor` which stream processor to start, use the value of the `Name` field specified in the call to `CreateStreamProcessor`.

Request Syntax

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

Request Parameters

The request accepts the following data in JSON format.

Name (p. 337)

The name of the stream processor to start processing.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 128.

Pattern: [a-zA-Z0-9_.\[-]+

Required: Yes

Response Elements

If the action is successful, the service sends back an HTTP 200 response with an empty HTTP body.

Errors

AccessDeniedException

You are not authorized to perform the action.

HTTP Status Code: 400

InternalServerError

Amazon Rekognition experienced a service issue. Try your call again.

HTTP Status Code: 500

InvalidArgumentException

Input parameter violated a constraint. Validate your parameter before calling the API operation again.

HTTP Status Code: 400

ProvisionedThroughputExceededException

The number of requests exceeded your throughput limit. If you want to increase this limit, contact Amazon Rekognition.

HTTP Status Code: 400

ResourceInUseException

HTTP Status Code: 400

ResourceNotFoundException

The collection specified in the request cannot be found.

HTTP Status Code: 400

ThrottlingException

Amazon Rekognition is temporarily unable to process the request. Try your call again.

HTTP Status Code: 500

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)

- [AWS SDK for .NET](#)

- [AWS SDK for C++](#)

- [AWS SDK for Go](#)

- [AWS SDK for Java](#)

- [AWS SDK for JavaScript](#)

- [AWS SDK for PHP V3](#)

- [AWS SDK for Python](#)

- [AWS SDK for Ruby V2](#)

(p. 337) 작업을 호출하여 비디오 처리를 시작합니다. 스트림 프로세서의 상태 정보를 얻으려면 [DescribeStreamProcessor](#) (p. 239)를 호출합니다. 호출할 수 있는 다른 작업으로는, 스트림 프로세서를 종지하기 위한 [StopStreamProcessor](#) (p. 339), 스트림 프로세서를 삭제하기 위한 [DeleteStreamProcessor](#) (p. 234) 등이 있습니다. 계정의 스트림 프로세서 목록을 확인하려면 [ListStreamProcessors](#) (p. 301)를 호출합니다.

스트림 프로세서가 실행되기 시작한 후, [CreateStreamProcessor](#)에서 지정한 Kinesis video stream을 통해 비디오를 Amazon Rekognition Video으로 스트리밍합니다. Kinesis Video Streams SDK [PutMedia](#) 작업을 사용하여 Kinesis video stream로 비디오를 전달합니다. 예제를 보려면 [PutMedia API 예제](#)를 참조하십시오.

애플리케이션이 Amazon Rekognition Video 분석 결과를 어떻게 사용하는지 알아보려면 [스트리밍 비디오 분석 결과 읽기](#) (p. 93) 단원을 참조하십시오.

Amazon Rekognition Video 스트림 프로세서 만들기

스트리밍 비디오를 분석하기 전에 Amazon Rekognition Video 스트림 프로세서를 만듭니다 ([CreateStreamProcessor](#) (p. 227)). 스트림 프로세서에는 Kinesis data stream과 Kinesis video stream에 대한 정보가 포함되어 있습니다. 또한 입력된 스트리밍 비디오에서 인식하고자 하는 얼굴이 포함된 모음의 식별자도 포함되어 있습니다. 스트림 프로세서의 이름도 지정합니다. 다음은 [CreateStreamProcessor](#) 요청에 대한 JSON 예제입니다.

```
{  
    "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
        }
    }
}
```

다음은 CreateStreamProcessor의 응답 예제입니다.

```
{
    "StreamProcessorArn": "arn:aws:rekognition:us-east-1:nnnnnnnnnnnn:streamprocessor/
    streamProcessorForCam"
}
```

Amazon Rekognition Video 스트림 프로세서 시작

CreateStreamProcessor에서 지정한 스트림 프로세서 이름으로 StartStreamProcessor (p. 337)를 호출하여 스트리밍 비디오 분석을 시작합니다. 다음은 StartStreamProcessor 요청에 대한 JSON 예제입니다.

```
{
    "Name": "streamProcessorForCam"
}
```

스트림 프로세서가 성공적으로 시작되면 JSON 본문이 비어 있는 상태로 HTTP 200 응답이 반환됩니다.

스트림 프로세서 사용

다음 예제 코드는 CreateStreamProcessor (p. 227), StartStreamProcessor (p. 337) 등, 다양한 스트림 프로세서 작업을 호출하는 방법을 보여줍니다. 이 예제에는 스트림 프로세서 작업을 호출하는 메서드를 제공하는 스트림 프로세서 관리자 클래스(StreamManager)가 포함됩니다. 스타터 클래스(Starter)는 StreamManager 객체를 생성하고 다양한 작업을 호출합니다.

예제를 구성하려면:

1. Starter 클래스 멤버 필드의 값을 원하는 값으로 설정합니다.
2. Starter 클래스 함수 main에서 원하는 함수 호출을 주석 처리합니다.

Starter 클래스

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

// Starter class. Use to create a StreamManager class
```

```
// and call stream processor operations.
package com.amazonaws.samples;
import com.amazonaws.samples.*;

public class Starter {

    public static void main(String[] args) {

        String streamProcessorName="Stream Processor Name";
        String kinesisVideoStreamArn="Kinesis Video Stream Arn";
        String kinesisDataStreamArn="Kinesis Data Stream Arn";
        String roleArn="Role Arn";
        String collectionId="Collection ID";
        Float matchThreshold=50F;

        try {
            StreamManager sm= new StreamManager(streamProcessorName,
                kinesisVideoStreamArn,
                kinesisDataStreamArn,
                roleArn,
                collectionId,
                matchThreshold);
            //sm.createStreamProcessor();
            //sm.startStreamProcessor();
            //sm.deleteStreamProcessor();
            //sm.deleteStreamProcessor();
            //sm.stopStreamProcessor();
            //sm.listStreamProcessors();
            //sm.describeStreamProcessor();
        }
        catch(Exception e){
            System.out.println(e.getMessage());
        }
    }
}
```

StreamManager 클래스

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

Amazon Rekognition Video으로 비디오 스트리밍

Amazon Rekognition Video으로 비디오를 스트리밍하려면 Amazon Kinesis Video Streams SDK를 사용해 Kinesis video stream을 만들고 사용합니다. PutMedia 작업은 Amazon Rekognition Video에서 사용하는

Kinesis 비디오 스트림에 비디오 데이터 조각을 씁니다. 비디오 데이터 조각 하나는 일반적으로 2–10초 길이이며, 독립적인 비디오 프레임 시퀀스를 포함합니다. Amazon Rekognition Video은 세 종류의 프레임(I, B 및 P)으로 구성될 수 있는 H.264 인코딩된 비디오를 지원합니다. 자세한 내용은 [Inter Frame](#)을 참조하십시오. 조각의 첫 번째 프레임은 I 프레임이어야 합니다. I-프레임은 다른 프레임과 별도로 디코딩될 수 있습니다.

비디오 데이터가 Kinesis video stream에 도착하면 Kinesis Video Streams가 조각에 고유 번호를 할당합니다. 예제를 보려면 [PutMedia API 예제](#)를 참조하십시오.

스트리밍 비디오 분석 결과 읽기

Amazon Kinesis Data Streams 클라이언트 라이브러리를 사용하여 Amazon Kinesis Data Streams 출력 스트림으로 전송되는 분석 결과를 사용할 수 있습니다. 자세한 내용은 [Reading Data from a Kinesis Data Stream](#)을 참조하십시오. Amazon Rekognition Video은 분석된 각 프레임에 대한 JSON 프레임 레코드를 Kinesis 출력 스트림에 저장합니다. Amazon Rekognition Video은 Kinesis video stream을 통해 전달 받은 모든 프레임을 분석하지 않습니다.

Kinesis data stream로 전송되는 프레임 레코드에는 프레임이 있는 Kinesis video stream 조각, 조각 내에서 프레임의 위치, 프레임에서 인식되는 얼굴에 대한 정보가 포함되어 있습니다. 또한 스트림 프로세서의 상태 정보도 포함됩니다. 자세한 내용은 [참조: Kinesis 얼굴 인식 레코드 \(p. 95\)](#) 단원을 참조하십시오.

Amazon Rekognition Video은 Amazon Rekognition Video 분석 정보를 Kinesis data stream로 스트리밍합니다. 다음은 단일 레코드에 대한 JSON 예제입니다.

```
{  
    "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": "noseTip"  
                    },  
                    {  
                        "X": 0.45960626,  
                        "Y": 0.4526856,  
                        "Type": "leftEar"  
                    },  
                    {  
                        "X": 0.45960626,  
                        "Y": 0.4526856,  
                        "Type": "rightEar"  
                    }  
                ]  
            }  
        }  
    ]  
}
```

```
        "Type": "nose"
    },
    {
        "X": 0.44958648,
        "Y": 0.4696949,
        "Type": "mouthLeft"
    },
    {
        "X": 0.46409217,
        "Y": 0.46704912,
        "Type": "mouthRight"
    }
],
"Pose": {
    "Pitch": 2.9691637,
    "Roll": -6.8904796,
    "Yaw": 23.84388
},
"Quality": {
    "Brightness": 40.592964,
    "Sharpness": 96.09616
}
},
"MatchedFaces": [
{
    "Similarity": 88.863960,
    "Face": {
        "BoundingBox": {
            "Height": 0.557692,
            "Width": 0.749838,
            "Left": 0.103426,
            "Top": 0.206731
        },
        "FaceId": "ed1b560f-d6af-5158-989a-ff586c931545",
        "Confidence": 99.999201,
        "ImageId": "70e09693-2114-57e1-807c-50b6d61fa4dc",
        "ExternalImageId": "matchedImage.jpeg"
    }
}
]
]
```

JSON 예제에서 다음 사항에 유의하십시오.

- **InputInformation** – Amazon Rekognition Video으로 비디오를 스트리밍하는 데 사용되는 Kinesis video stream 관련 정보입니다. 자세한 내용은 [InputInformation \(p. 97\)](#) 단원을 참조하십시오.
- **StreamProcessorInformation** – Amazon Rekognition Video 스트림 프로세서의 상태 정보입니다. **Status** 필드에 입력할 수 있는 값은 RUNNING뿐입니다. 자세한 내용은 [StreamProcessorInformation \(p. 97\)](#) 단원을 참조하십시오.
- **FaceSearchResponse** – 스트리밍 비디오에서 입력 모음의 얼굴과 일치하는 얼굴에 대한 정보가 들어 있습니다. [FaceSearchResponse \(p. 98\)](#)에는 분석한 비디오 프레임에서 감지되었던 얼굴, 즉 [DetectedFace \(p. 98\)](#) 객체가 포함되어 있습니다. **MatchedFaces** 배열에는 감지된 얼굴별로 입력 모음에서 발견된 일치하는 얼굴 객체([MatchedFace \(p. 99\)](#))와 유사성 점수가 나와 있습니다.

Kinesis Video Stream을 Kinesis Data Stream으로 매핑

Kinesis video stream 프레임을 Kinesis data stream로 전송되는 분석 프레임으로 매핑할 수 있습니다. 예를 들어 스트리밍 동영상이 표시되는 도중에도 인식된 사람들의 얼굴 주위로 상자를 표시하는 것이 가능합니다. 경계 상자 좌표는 Kinesis 얼굴 인식 레코드에 포함되어 Kinesis data stream으로 전송됩니다. 경계 상자를 정

확하게 표시하려면 Kinesis 얼굴 인식 레코드와 함께 전송된 시간 정보를 원본 Kinesis video stream의 해당 프레임과 매핑시켜야 합니다.

Kinesis video stream을 Kinesis data stream에 매핑할 때 사용하는 기법은 라이브 미디어(라이브 스트리밍 동영상 등)를 스트리밍하는 경우 또는 아카이브 미디어(저장 동영상)를 스트리밍하는 경우에 따라 달라집니다.

라이브 미디어를 스트리밍하는 경우의 매핑

Kinesis video stream 프레임을 Kinesis data stream 프레임으로 매핑하려면

1. [PutMedia](#) 작업의 입력 파라미터 `FragmentTimeCodeType`을 `RELATIVE`로 설정합니다.
2. [PutMedia](#)를 호출하여 라이브 미디어를 Kinesis video stream으로 전송합니다.
3. Kinesis data stream에서 Kinesis 얼굴 인식 레코드가 수신되면 [KinesisVideo \(p. 97\)](#) 필드의 `ProducerTimestamp` 값과 `FrameOffsetInSeconds` 필드 값을 저장합니다.
4. `ProducerTimestamp` 필드 값과 `FrameOffsetInSeconds` 필드 값을 합산하여 Kinesis video stream 프레임과 일치하는 타임스탬프를 계산합니다.

아카이브 미디어를 스트리밍하는 경우의 매핑

Kinesis video stream 프레임을 Kinesis data stream 프레임으로 매핑하려면

1. [PutMedia](#)를 호출하여 아카이브 미디어를 Kinesis video stream으로 전송합니다.
2. [PutMedia](#) 작업 응답에서 `Acknowledgement` 객체가 수신되면 `Payload` 필드의 `FragmentNumber` 필드 값을 저장합니다. 여기에서 `FragmentNumber`는 MKV 클러스터의 조각 수입니다.
3. Kinesis data stream에서 Kinesis 얼굴 인식 레코드가 수신되면 [KinesisVideo \(p. 97\)](#) 필드의 `FrameOffsetInSeconds` 필드 값을 저장합니다.
4. 2단계와 3단계에서 저장한 `FrameOffsetInSeconds` 및 `FragmentNumber` 값을 사용하여 매핑을 계산합니다. 여기에서 `FrameOffsetInSeconds`는 Amazon Kinesis data stream으로 전송되는 특정 `FragmentNumber`와 함께 조각에 대한 오프셋입니다. 임의의 조각 수에 대한 동영상 프레임을 가져오는 방법에 대한 자세한 내용은 [Amazon Kinesis Video Streams 아카이브 미디어](#) 단원을 참조하십시오.

참조: Kinesis 얼굴 인식 레코드

Amazon Rekognition Video은 스트리밍 비디오에서 얼굴을 인식할 수 있습니다. 분석된 각 프레임에 대해 Amazon Rekognition Video은 JSON 프레임 레코드를 Kinesis 데이터 스트림으로 출력합니다. Amazon Rekognition Video은 Kinesis video stream을 통해 전달 받은 모든 프레임을 분석하지 않습니다.

JSON 프레임 레코드에는 입력 및 출력 스트림, 스트림 프로세서의 상태, 분석된 프레임에서 인식되는 얼굴에 대한 정보가 들어 있습니다. 이 단원에서는 JSON 프레임 레코드에 대한 참조 정보를 제시합니다.

다음은 Kinesis data stream 레코드에 대한 JSON 구문입니다. 자세한 내용은 [스트리밍 비디오 작업 \(p. 84\)](#) 단원을 참조하십시오.

```
{  
    "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": [
                    {
                        "Face": {
                            "BoundingBox": {
                                "Width": number,
                                "Top": number,
                                "Height": number,
                                "Left": number
                            },
                            "Confidence": number,
                            "ExternalImageId": "string",
                            "FaceId": "string",
                            "ImageId": "string"
                        },
                        "Similarity": number
                    }
                ]
            }
        ]
    }
}
```

JSON 레코드

JSON 레코드에는 Amazon Rekognition Video에서 처리하는 프레임에 대한 정보가 포함되어 있습니다. 레코드에는 스트리밍 비디오, 분석된 프레임의 상태, 프레임에서 인식되는 얼굴에 대한 정보가 포함됩니다.

InputInformation

Amazon Rekognition Video으로 비디오를 스트리밍하는 데 사용되는 Kinesis video stream 관련 정보입니다.

유형: [InputInformation \(p. 97\)](#) 객체

StreamProcessorInformation

Amazon Rekognition Video 스트림 프로세서에 대한 정보입니다. 여기에는 스트림 프로세서의 현재 상태에 대한 상태 정보가 포함됩니다.

유형: [StreamProcessorInformation \(p. 97\)](#) 객체

FaceSearchResponse

스트리밍 비디오 프레임에서 감지된 얼굴과 입력 모음에서 발견된 일치하는 얼굴에 대한 정보입니다.

유형: [FaceSearchResponse \(p. 98\)](#) 객체 배열

InputInformation

Amazon Rekognition Video에서 사용하는 소스 비디오 스트림에 대한 정보입니다. 자세한 내용은 [스트리밍 비디오 작업 \(p. 84\)](#) 단원을 참조하십시오.

KinesisVideo

유형: [KinesisVideo \(p. 97\)](#) 객체

KinesisVideo

Amazon Rekognition Video으로 소스 비디오를 스트리밍하는 Kinesis video stream 관련 정보입니다. 자세한 내용은 [스트리밍 비디오 작업 \(p. 84\)](#) 단원을 참조하십시오.

StreamArn

Kinesis video stream의 Amazon Resource Name(ARN)입니다.

유형: 문자열

FragmentNumber

이 레코드가 나타내는 프레임이 포함된 스트리밍 비디오 조각입니다.

유형: 문자열

ProducerTimestamp

조각의 생산자 측 Unix 타임스탬프입니다. 자세한 내용은 [PutMedia](#) 단원을 참조하십시오.

형식: 숫자

ServerTimestamp

조각의 서버 측 Unix 타임스탬프입니다. 자세한 내용은 [PutMedia](#) 단원을 참조하십시오.

형식: 숫자

FrameOffsetInSeconds

조각 내의 프레임 오프셋(초 단위)입니다.

형식: 숫자

StreamProcessorInformation

스트림 프로세서에 대한 상태 정보입니다.

상태

스트림 프로세서의 현재 상태입니다. 가능한 값은 RUNNING입니다.

유형: 문자열

FaceSearchResponse

스트리밍 비디오 프레임에서 감지된 얼굴과 모음에서 발견된 그것과 일치하는 얼굴에 대한 정보입니다. [CreateStreamProcessor \(p. 227\)](#) 호출에 모음을 지정합니다. 자세한 내용은 [스트리밍 비디오 작업 \(p. 84\)](#) 단원을 참조하십시오.

DetectedFace

분석된 비디오 프레임에서 감지된 얼굴의 세부 정보입니다.

유형: [DetectedFace \(p. 98\)](#) 객체

MatchedFaces

[DetectedFace](#)에서 감지된 얼굴과 일치하는 얼굴 모음의 세부 정보 배열입니다.

유형: [MatchedFace \(p. 99\)](#) 객체 배열

DetectedFace

스트리밍 비디오 프레임에서 감지된 얼굴에 대한 정보입니다. 입력 모음에 들어 있는 일치하는 얼굴은 [MatchedFace \(p. 99\)](#) 객체 필드에서 사용할 수 있습니다.

BoundingBox

분석된 비디오 프레임 내에서 감지된 얼굴의 경계 상자 좌표입니다. [BoundingBox](#) 객체는 이미지 분석에 사용되는 [BoundingBox](#) 객체와 동일한 속성을 가집니다.

유형: [BoundingBox \(p. 344\)](#) 객체

신뢰도

감지된 얼굴이 실제 얼굴일 가능성의 의미하는 Amazon Rekognition Video의 신뢰도 수준입니다(1-100). 1은 신뢰도가 가장 낮고, 100은 가장 높습니다.

형식: 숫자

표식

얼굴 표식의 배열입니다.

유형: [Landmark \(p. 376\)](#) 객체 배열

포즈

피치, 률 및 요로 판단되는 얼굴의 포즈를 나타냅니다.

유형: [Pose \(p. 386\)](#) 객체

화질

얼굴 이미지의 밝기와 선명도를 나타냅니다.

유형: [ImageQuality \(p. 369\)](#) 객체

MatchedFace

분석된 비디오 프레임에서 감지된 얼굴과 일치하는 얼굴에 대한 정보입니다.

얼굴

[DetectedFace \(p. 98\)](#) 객체의 얼굴과 일치하는, 입력 모음의 얼굴에 대한 얼굴 일치 정보입니다.

유형: [Face \(p. 357\)](#) 객체

유사성

얼굴이 일치하는 신뢰도 수준입니다(1-100). 1은 신뢰도가 가장 낮고, 100은 가장 높습니다.

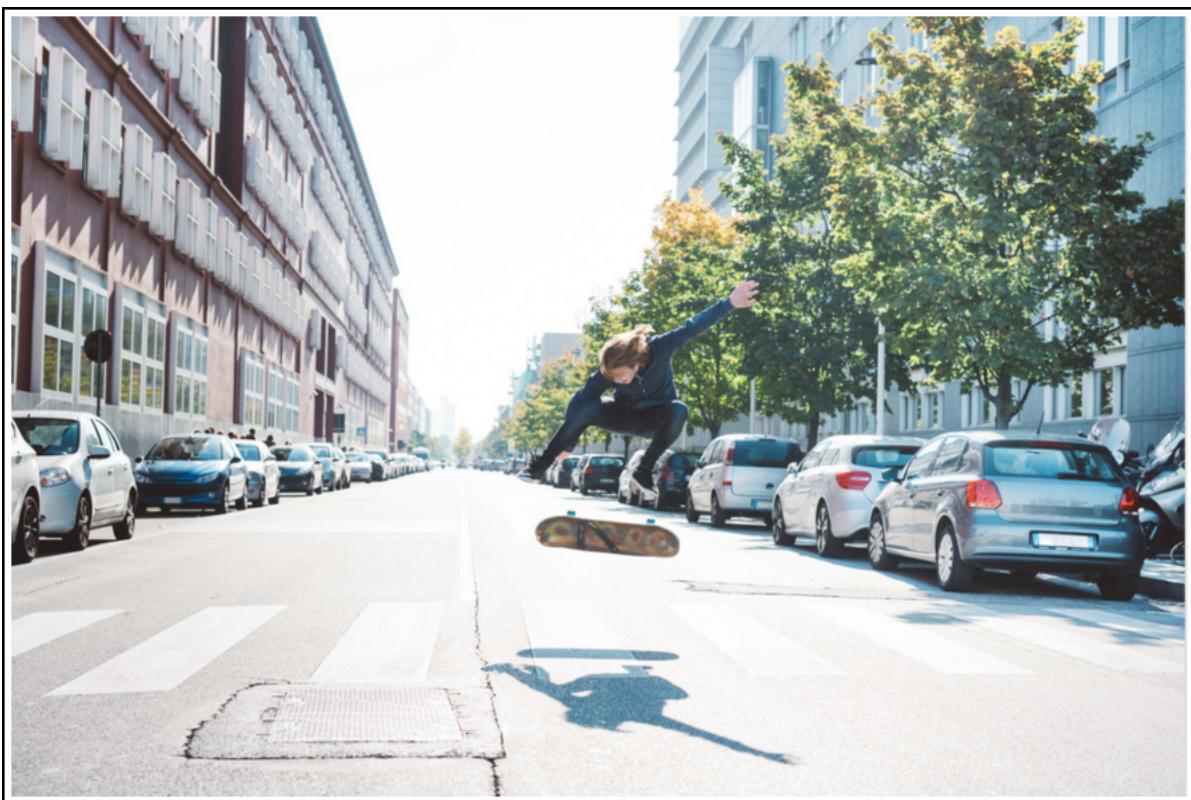
형식: 숫자

객체 및 장면 감지

이 섹션에서는 Amazon Rekognition Image 및 Amazon Rekognition Video를 통해 이미지와 비디오에서의 레이블 감지에 대한 정보를 다룹니다.

레이블 또는 태그는 콘텐츠를 기반으로 이미지나 비디오에서 발견되는 객체, 장면 또는 개념을 가리킵니다. 예를 들면 열대 지역 해변에 있는 사람을 찍은 사진에는 사람, 물, 모래, 야자수 및 수영복(객체), 해변(장면), 야외(개념) 같은 레이블이 포함될 수 있습니다. Amazon Rekognition Video도 스키 타기나 자전거 타기와 같은 활동을 감지합니다. Amazon Rekognition Image는 이미지에서 활동을 감지하지 않습니다.

예를 들어 다음 이미지에서 Amazon Rekognition Image은 사람의 유무, 스케이트보드, 주차된 차량 등 다양한 정보를 감지할 수 있습니다. Amazon Rekognition Video 및 Amazon Rekognition Image는 또한 Amazon Rekognition이 감지된 각 레이블의 정확도 면에서 얼마나 큰 신뢰도를 갖는지에 대한 백분율 점수도 제공합니다.



항목

- [이미지에서 레이블 감지 \(p. 100\)](#)
- [비디오에서 레이블 감지 \(p. 102\)](#)

이미지에서 레이블 감지

DetectLabels (p. 247) 작업을 사용하여 이미지에서 레이블을 감지할 수 있습니다. 관련 예제는 [Amazon S3 버킷에 저장된 이미지 분석 \(p. 37\)](#) 단원을 참조하십시오.

DetectLabels 작업 요청

DetectLabel에 대한 입력은 이미지입니다. 이 예제 JSON 입력에서는 Amazon S3 버킷에서 소스 이미지를 불러옵니다. MaxLabels는 응답에 반환되는 레이블의 최대 수입니다. MinConfidence는 Amazon Rekognition Image가 응답에 반환하기 위해 충족해야 할 감지된 레이블의 최소 정확성 신뢰도 수준입니다.

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

DetectLabels 응답

DetectLabels의 응답은 이미지에서 감지된 레이블의 배열과 해당 신뢰도 수준입니다.

다음은 DetectLabels의 응답 예제입니다.

```
{  
    "Labels": [  
        {  
            "Confidence": 98.4629,  
            "Name": "beacon"  
        },  
        {  
            "Confidence": 98.4629,  
            "Name": "building"  
        },  
        {  
            "Confidence": 98.4629,  
            "Name": "lighthouse"  
        },  
        {  
            "Confidence": 87.7924,  
            "Name": "rock"  
        },  
        {  
            "Confidence": 68.1049,  
            "Name": "sea"  
        }  
    ]  
}
```

이 응답은 작업에서 5개 레이블(즉, 비콘, 건물, 등대, 바위, 바다)을 감지했음을 보여 줍니다. 각 레이블에는 연결된 신뢰도 수준이 있습니다. 예를 들어 감지 알고리즘 상 이미지에 건물이 포함될 신뢰도는 98.4629%입니다.

제공한 입력 이미지에 사람이 포함된 경우, DetectLabels 작업은 다음 응답 예에서 보듯 사람, 옷, 정장, 셀프 카메라 등의 레이블을 감지합니다.

```
{  
    "Labels": [  
        {  
            "Confidence": 99.2786,  
            "Name": "person"  
        }  
    ]  
}
```

```
        },
        {
            "Confidence": 90.6659,
            "Name": "clothing"
        },
        {
            "Confidence": 90.6659,
            "Name": "suit"
        },
        {
            "Confidence": 70.0364,
            "Name": "selfie"
        }
    ]
}
```

Note

이미지의 얼굴을 기술하는 얼굴 특징을 원하는 경우, `DetectFaces` 작업을 대신 사용합니다.

비디오에서 레이블 감지

Amazon Rekognition Video는 비디오에서 레이블을 감지하고 레이블이 감지된 시간을 감지할 수 있습니다. SDK 코드에 대한 예는 [Java](#) 또는 [Python](#)으로 Amazon S3 버킷에 저장된 비디오 분석(SDK) (p. 66) 단원을 참조하십시오. AWS CLI 예제는 [AWS Command Line Interface](#)로 비디오 분석 (p. 72) 단원을 참조하십시오.

Amazon Rekognition Video 레이블 감지은 비동기식 작업입니다. 비디오에서 레이블 감지를 시작하려면 [StartLabelDetection](#) (p. 330)을 호출합니다. Amazon Rekognition Video는 비디오 분석의 완료 상태를 Amazon Simple Notification Service 주제에 게시합니다. 비디오 분석이 성공적으로 완료되면 [GetLabelDetection](#) (p. 278)을 호출하여 감지된 레이블을 가져오십시오. 비디오 인식 API 작업을 호출하는 방법에 대한 자세한 내용은 [Amazon Rekognition Video 작업 불러오기](#) (p. 59) 단원을 참조하십시오.

GetLabelDetection 작업 응답

`GetLabelDetection`은 비디오에서 감지된 레이블에 대한 정보가 포함된 배열(`Labels`)을 반환합니다. 이 배열을 감지된 레이블 또는 시간별로 정렬하려면 `SortBy` 파라미터를 지정합니다.

다음은 `GetLabelDetection`의 JSON 응답 예입니다. 응답에서 다음에 유의하십시오.

- 정렬 순서 – 반환된 레이블의 배열이 시간별로 정렬됩니다. 레이블별로 정렬하려면, `GetLabelDetection`에서 `SortBy` 입력 파라미터에 `NAME`을 지정하십시오. If the label appears multiple times in the video, there will be multiples instances of the ([LabelDetection](#) (p. 375)) element.
- 레이블 정보 – `LabelDetection` 배열 요소에는 레이블 이름과, 감지된 레이블의 정확도 측면에서 Amazon Rekognition이 제공하는 신뢰도를 포함하는 ([Label](#) (p. 373)) 객체가 포함되어 있습니다. `Timestamp`은 레이블이 감지될 때 비디오의 시작 시간(밀리초)입니다.
- 페이지 정보 – 예제는 레이블 감지 정보의 페이지 하나를 보여줍니다. `GetLabelDetection`의 `MaxResults` 입력 파라미터에서 반환하는 `LabelDetection` 객체의 개수를 지정할 수 있습니다. `MaxResults` 보다 많은 결과가 존재할 경우 `GetLabelDetection`은 결과의 다음 페이지를 가져올 때 사용되는 토큰(`NextToken`)을 반환합니다. 자세한 내용은 [Amazon Rekognition Video 분석 결과 가져오기](#) (p. 61) 단원을 참조하십시오.
- 비디오 정보 – `GetLabelDetection`이 반환하는 정보의 각 페이지에 있는 비디오 형식 (`VideoMetadata`)에 관한 정가 응답에 포함됩니다.

```
{
```

```
"Labels": [{"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"}, "Timestamp": 0}, {"Label": {"Confidence": 71.44749450683594, "Name": "Clothing"}, "Timestamp": 0}, {"Label": {"Confidence": 71.44749450683594, "Name": "Overcoat"}, "Timestamp": 0}, {"Label": {"Confidence": 71.44749450683594, "Name": "Suit"}, "Timestamp": 0}, {"Label": {"Confidence": 58.84939956665039, "Name": "Jar"}, "Timestamp": 0}, {"Label": {"Confidence": 58.84939956665039, "Name": "Porcelain"}, "Timestamp": 0}, {"Label": {"Confidence": 58.84939956665039, "Name": "Vase"}, "Timestamp": 0}], "NextToken": "BgPXS37fNUY+Hrd7jYbBoTk1I5LevLm/MV+dhCwbXxoVOgMi0di6xYSel6/Dztya/Pflx9xxxx=", "VideoMetadata": {"Codec": "h264", "DurationMillis": 67301},
```

```
"Format": "mov,mp4,m4a,3gp,3g2,mj2",
"FrameHeight": 1080,
"FrameRate": 29.970029830932617,
"FrameWidth": 1920
}
}
```

얼굴 감지 및 분석

Amazon Rekognition은 이미지와 비디오에서 얼굴을 감지할 수 있습니다. 이 단원에서는 얼굴 분석을 위한 비-스토리지 작업을 다룹니다. Amazon Rekognition을 통해 이미지에서 얼굴을 감지한 위치에 대한 정보뿐 아니라 눈의 위치와 같은 얼굴 표식과, 행복해 보이거나 슬퍼 보이는 등의 감지된 감정에 대한 정보를 가져올 수 있습니다. 원본 이미지의 한 얼굴과 다른 이미지에서 감지된 얼굴도 비교할 수 있습니다. 비디오 분석을 통해 비디오 전체에서 얼굴이 감지되는 시점을 추적할 수 있습니다.

예를 들어, 얼굴이 포함되어 있는 이미지를 제공하면 Amazon Rekognition은 이미지에서 얼굴을 감지하고, 얼굴의 얼굴 속성을 분석한 다음, 이미지에서 감지된 얼굴과 얼굴 속성의 백분율 신뢰 점수를 반환합니다.



이 단원에서는 이미지와 비디오의 얼굴 분석 예를 제공합니다. Amazon Rekognition API 명령 사용에 대한 자세한 내용은 [이미지 작업 \(p. 35\)](#) 및 [저장된 비디오 작업 \(p. 57\)](#) 단원을 참조하십시오.

스토리지 작업을 사용하여 이미지에서 감지된 얼굴 메타데이터를 저장할 수 있습니다. 나중에 이미지와 비디오에 저장된 얼굴을 검색할 수 있습니다. 예를 들어, 이것을 이용해 비디오에서 특정 인물을 검색할 수 있습니다. 자세한 내용은 [모음에서 얼굴 검색 \(p. 127\)](#) 단원을 참조하십시오.

항목

- [이미지에서 얼굴 감지 \(p. 106\)](#)
- [이미지에 있는 얼굴 비교 \(p. 114\)](#)
- [저장된 비디오에서 얼굴 감지 \(p. 121\)](#)

이미지에서 얼굴 감지

Amazon Rekognition Image는 입력 이미지에서 눈, 코, 입 등 주요 얼굴 특징을 찾아 얼굴을 감지하는 [DetectFaces \(p. 243\)](#) 작업을 제공합니다. Amazon Rekognition Image는 이미지에서 가장 큰 얼굴 100개를 감지합니다.

입력 이미지를 이미지 바이트 배열(base64 인코딩 이미지 바이트)로 제공하거나 Amazon S3 객체를 지정할 수 있습니다. 이 절차에서는 S3 버킷에 이미지(JPEG 또는 PNG)를 업로드하고 객체 키 이름을 지정합니다.

이미지에서 얼굴을 감지하려면

1. 아직 설정하지 않았다면 다음과 같이 하십시오.
 - a. AmazonRekognitionFullAccess 권한과 AmazonS3ReadOnlyAccess 권한을 가진 IAM 사용자를 만들어가 업데이트합니다. 자세한 내용은 [1단계: AWS 계정 설정 및 IAM 사용자 만들기 \(p. 11\)](#) 단원을 참조하십시오.
 - b. AWS CLI와 AWS SDK를 설치하고 구성합니다. 자세한 내용은 [2단계: AWS CLI 및 AWS SDK 설정 \(p. 11\)](#) 단원을 참조하십시오.
2. 한 개 이상의 얼굴이 포함된 이미지를 S3 버킷에 업로드합니다.

이에 관한 지침은 Amazon Simple Storage Service 콘솔 사용 설명서의 [Amazon S3에 객체 업로드](#)를 참조하십시오.

3. 다음 예제를 사용하여 DetectFaces를 호출합니다.

Java

이 예제에서는 감지된 얼굴의 추정 연령 범위를 표시하고 감지된 모든 얼굴 속성의 JSON을 나열합니다. photo의 값을 이미지 파일 이름으로 변경합니다. bucket의 값을 이미지가 저장된 Amazon S3 버킷으로 변경합니다.

```
//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();
}

}
```

AWS CLI

이 예제는 detect-faces AWS CLI 작업의 JSON 출력을 표시합니다. `file`을 이미지 파일의 이름으로 바꿉니다. `bucket`을 이미지 파일이 들어 있는 Amazon S3 버킷의 이름으로 바꿉니다.

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

Python

이 예제에서는 감지된 얼굴의 추정 연령 범위를 표시하고 감지된 모든 얼굴 속성의 JSON을 나열합니다. `photo`의 값을 이미지 파일 이름으로 변경합니다. `bucket`의 값을 이미지가 저장된 Amazon S3 버킷으로 변경합니다.

```
#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
```

```
if __name__ == "__main__":
    photo='input.jpg'
    bucket='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))
```

.NET

이 예제에서는 감지된 얼굴의 추정 연령 범위를 표시하고 감지된 모든 얼굴 속성의 JSON을 나열합니다. photo의 값을 이미지 파일 이름으로 변경합니다. bucket의 값을 이미지가 저장된 Amazon S3 버킷으로 변경합니다.

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

DetectFaces 작업 요청

DetectFaces에 대한 입력은 이미지입니다. 이 예제에서는 Amazon S3 버킷에서 이미지를 불러옵니다. Attributes 파라미터는 모든 얼굴 속성을 반환하도록 지정합니다. 자세한 내용은 [이미지 작업 \(p. 35\)](#) 단원을 참조하십시오.

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

DetectFaces 작업 응답

DetectFaces는 감지된 각 얼굴에 대해 다음 정보를 반환합니다.

- 경계 상자 – 얼굴 주위의 경계 상자의 좌표. 자세한 내용은 [이미지 방향 및 경계 상자 좌표 가져오기 \(p. 47\)](#) 단원을 참조하십시오.
- 신뢰도 – 경계 상자에 얼굴이 포함될 신뢰도 수준.
- 얼굴 표식 – 얼굴 표식의 배열. 응답은 왼쪽 눈, 오른쪽 눈, 입 같은 각각의 표식의 x, y 좌표를 제공합니다.
- 얼굴 속성 – 성별 또는 얼굴에 턱수염이 있는지 여부 등 일련의 얼굴 속성. 응답은 각 속성의 값을 제공합니다. 이 값은 부울(사람이 선글라스를 착용하고 있는지), 문자열(남성인지 여성인지) 등 다양한 형식이 될 수 있습니다. 또한 대부분의 속성의 경우, 응답은 해당 속성에 대해 감지된 값의 신뢰도도 제공합니다.
- 품질 – 얼굴의 밝기와 선명도를 기술합니다. 최상의 얼굴 감지를 보장하는 것에 관한 내용은 [얼굴 인식 입력 이미지에 대한 권장 사항 \(p. 36\)](#) 단원을 참조하십시오.
- 포즈 – 이미지 내 얼굴의 회전을 기술합니다.
- 감정 – 분석에서 신뢰도가 포함된 일련의 감정.

다음은 DetectFaces API 호출 응답의 예입니다.

```
{  
    "FaceDetails": [  
        {  
            "AgeRange": {  
                "High": 36,  
                "Low": 19  
            },  
            "Beard": {  
                "Confidence": 99.99388122558594,  
                "Value": false  
            },  
            "BoundingBox": {  
                "Height": 0.35353535413742065,  
                "Left": 0.10707070678472519,  
                "Top": 0.1277778506278992,  
                "Width": 0.47138047218322754  
            },  
            "Confidence": 99.99736785888672,  
            "Emotions": [  
                {  
                    "Confidence": 89.59288024902344,  
                    "Type": "HAPPY"  
                },  
                {  
                    "Confidence": 5.3619384765625,  
                    "Type": "CALM"  
                },  
                {  
                    "Confidence": 4.1074934005737305,  
                    "Type": "ANGRY"  
                }  
            ],  
            "Eyeglasses": {  
                "Confidence": 99.96102142333984,  
                "Value": false  
            },  
            "EyesOpen": {  
                "Confidence": 99.97982788085938,  
                "Value": true  
            },  
            "Gender": {  
                "Confidence": 100,  
                "Value": "Female"  
            },  
            "Landmarks": [  
                {  
                    "Type": "eyeLeft",  
                    "X": 0.23772846162319183,  
                    "Y": 0.2778792679309845  
                },  
                {  
                    "Type": "eyeRight",  
                    "X": 0.3944779932498932,  
                    "Y": 0.2527812421321869  
                },  
                {  
                    "Type": "nose",  
                    "X": 0.28617316484451294,  
                    "Y": 0.3367626965045929  
                },  
                {  
                    "Type": "mouthLeft",  
                    "X": 0.288897842168808,  
                    "Y": 0.2527812421321869  
                }  
            ]  
        }  
    ]  
}
```

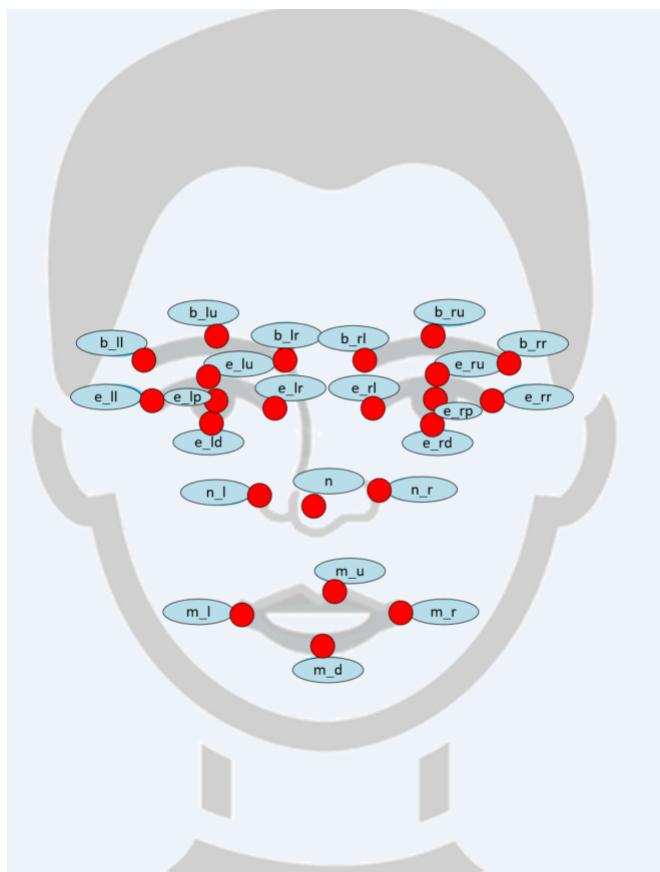
```
        "Y": 0.4019121527671814
    },
    {
        "Type": "mouthRight",
        "X": 0.41363197565078735,
        "Y": 0.38766127824783325
    },
    {
        "Type": "leftPupil",
        "X": 0.25244733691215515,
        "Y": 0.27739065885543823
    },
    {
        "Type": "rightPupil",
        "X": 0.4029206931591034,
        "Y": 0.24940147995948792
    },
    {
        "Type": "leftEyeBrowLeft",
        "X": 0.17436790466308594,
        "Y": 0.24362699687480927
    },
    {
        "Type": "leftEyeBrowUp",
        "X": 0.21201953291893005,
        "Y": 0.2338741421699524
    },
    {
        "Type": "leftEyeBrowRight",
        "X": 0.2513192892074585,
        "Y": 0.24069637060165405
    },
    {
        "Type": "rightEyeBrowLeft",
        "X": 0.32193484902381897,
        "Y": 0.21918891370296478
    },
    {
        "Type": "rightEyeBrowUp",
        "X": 0.38548189401626587,
        "Y": 0.19604144990444183
    },
    {
        "Type": "rightEyeBrowRight",
        "X": 0.45430734753608704,
        "Y": 0.2027731090784073
    },
    {
        "Type": "leftEyeLeft",
        "X": 0.20944036543369293,
        "Y": 0.28281378746032715
    },
    {
        "Type": "leftEyeRight",
        "X": 0.2710888683795929,
        "Y": 0.2738289535045624
    },
    {
        "Type": "leftEyeUp",
        "X": 0.2339935451745987,
        "Y": 0.2687133252620697
    },
    {
        "Type": "leftEyeDown",
        "X": 0.23892717063426971,
        "Y": 0.28660306334495544
```

```
        },
        {
            "Type": "rightEyeLeft",
            "X": 0.36334219574928284,
            "Y": 0.2598327100276947
        },
        {
            "Type": "rightEyeRight",
            "X": 0.4293186664581299,
            "Y": 0.249033123254776
        },
        {
            "Type": "rightEyeUp",
            "X": 0.39038628339767456,
            "Y": 0.2431529313325882
        },
        {
            "Type": "rightEyeDown",
            "X": 0.3967171609401703,
            "Y": 0.26075780391693115
        },
        {
            "Type": "noseLeft",
            "X": 0.28841185569763184,
            "Y": 0.3598580062389374
        },
        {
            "Type": "noseRight",
            "X": 0.3451237976551056,
            "Y": 0.3516968786716461
        },
        {
            "Type": "mouthUp",
            "X": 0.3349839448928833,
            "Y": 0.38809144496917725
        },
        {
            "Type": "mouthDown",
            "X": 0.3422594964504242,
            "Y": 0.41868656873703003
        }
    ],
    "MouthOpen": {
        "Confidence": 99.97990417480469,
        "Value": false
    },
    "Mustache": {
        "Confidence": 99.97885131835938,
        "Value": false
    },
    "Pose": {
        "Pitch": 1.0711474418640137,
        "Roll": -10.933034896850586,
        "Yaw": -25.838171005249023
    },
    "Quality": {
        "Brightness": 43.86729431152344,
        "Sharpness": 99.95819854736328
    },
    "Smile": {
        "Confidence": 96.89310455322266,
        "Value": true
    },
    "Sunglasses": {
        "Confidence": 80.4033432006836,
        "Value": false
    }
}
```

```
        }  
    ]  
}
```

다음을 참조하십시오.

- Pose 데이터는 감지된 얼굴의 회전을 기술합니다. BoundingBox와 Pose 데이터의 조합을 사용하여 애플리케이션이 표시하는 얼굴 주위에 경계 상자를 그릴 수 있습니다.
- Quality는 얼굴의 밝기와 선명도를 기술합니다. 이는 여러 이미지에서 얼굴을 비교해 가장 좋은 얼굴을 찾는데 유용할 수 있습니다.
- DetectFaces 작업은 먼저 입력 이미지의 방향을 감지한 다음 얼굴 특징을 감지합니다. 응답의 OrientationCorrection 작업은 감지된 회전 각도를 반환합니다(시계 반대 방향). 애플리케이션은 이 이미지를 표시할 때 이 값을 사용하여 이미지 방향을 수정합니다.
- 이전 응답은 서비스가 감지할 수 있는 모든 얼굴 landmarks, 모든 얼굴 속성 및 감정을 보여 줍니다. 응답에서 이 모두를 얻으려면 attributes 파라미터를 값 ALL로 지정해야 합니다. DetectFaces API는 기본적으로 BoundingBox, Confidence, Pose, Quality, landmarks와 같은 5가지 얼굴 속성만 반환합니다. 기본 표식은 eyeLeft, eyeRight, nose, mouthLeft, mouthRight가 반환됩니다.
- 다음 그림은 DetectFaces API 작업이 반환하는 얼굴에서 얼굴 표식(Landmarks)의 상대적 위치를 보여 줍니다.



이미지의 레이블은 다음 표에 나와 있는 대로 [Landmark](#) (p. 376) 데이터 유형으로 매핑됩니다. 이해를 돋기 위해 왼쪽 눈과 오른쪽 눈 표식 점은 이미지에 표시하지 않았습니다.

레이블	표식
b_ll	leftEyeBrowLeft
b_lu	leftEyeBrowUp
b_lr	leftEyeBrowRight
e_ll	leftEyeLeft
e_lu	leftEyeUp
e_lr	leftEyeRight
e_ld	leftEyeDown
e_lp	leftPupil
n_l	noseLeft
n	nose
n_r	noseRight
m_u	mouthUp
m_d	mouthDown
m_l	mouthLeft
m_r	mouthRight
b_rl	rightEyeBrowLeft
b_ru	rightEyeBrowUp
b_rr	rightEyeBrowRight
e_rl	rightEyeLeft
e_ru	rightEyeUp
e_rr	rightEyeRight
e_rp	rightPupil
e_rd	rightEyeDown

이미지에 있는 얼굴 비교

소스 이미지에 있는 얼굴을 대상 이미지에 있는 각 얼굴과 비교하려면 [CompareFaces \(p. 219\)](#) 작업을 사용합니다.

응답에서 반환되기를 원하는 일치에서 최소 신뢰도를 지정하려면 요청에서 `similarityThreshold`를 사용합니다. 자세한 내용은 [CompareFaces \(p. 219\)](#) 단원을 참조하십시오.

여러 얼굴이 포함된 원본 이미지를 제공하는 경우, 서비스는 원본 이미지에서 가장 큰 얼굴을 감지하여 이를 사용해 대상 이미지에서 감지된 각각의 얼굴과 비교합니다.

원본 이미지와 대상 이미지를 이미지 바이트 배열(base64 인코딩 이미지)로 제공하거나 Amazon S3 객체를 지정할 수 있습니다. AWS CLI 예제에서는 2개의 JPEG 이미지를 Amazon S3 버킷에 업로드하고 객체 키 이

름을 지정합니다. 다른 예제에서는 로컬 파일 시스템에서 파일 2개를 로드하여 이미지 바이트 배열로 입력합니다.

얼굴을 비교하려면

1. 아직 설정하지 않았다면 다음과 같이 하십시오.
 - a. AmazonRekognitionFullAccess 권한과 AmazonS3ReadOnlyAccess(AWS CLI 예제만 해당) 권한을 가진 IAM 사용자를 만들어가 업데이트합니다. 자세한 내용은 [1단계: AWS 계정 설정 및 IAM 사용자 만들기 \(p. 11\)](#) 단원을 참조하십시오.
 - b. AWS CLI와 AWS SDK를 설치하고 구성합니다. 자세한 내용은 [2단계: AWS CLI 및 AWS SDK 설정 \(p. 11\)](#) 단원을 참조하십시오.
2. 다음 예제 코드를 사용하여 `CompareFaces` 작업을 호출하십시오.

Java

이 예제는 로컬 파일 시스템에서 불러오는 소스 이미지와 대상 이미지의 일치 얼굴에 대한 정보를 표시합니다.

`sourceImage` 값과 `targetImage` 값을 소스 및 대상 이미지의 파일 이름과 경로로 바꿉니다.

```
//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 " + face.getConfidence().toString()
            + "% confidence.");

    }
    List<ComparedFace> uncompered = compareFacesResult.getUnmatchedFaces();

    System.out.println("There was " + uncompered.size()
        + " face(s) that did not match");
    System.out.println("Source image rotation: " +
compareFacesResult.getSourceImageOrientationCorrection());
    System.out.println("target image rotation: " +
compareFacesResult.getTargetImageOrientationCorrection());
}
}
```

AWS CLI

이 예제는 compare-faces AWS CLI 작업의 JSON 출력을 표시합니다.

bucket-name을 소스 이미지와 대상 이미지가 들어 있는 Amazon S3 버킷의 이름으로 바꿉니다.
source.jpg와 target.jpg를 소스 이미지와 대상 이미지의 파일 이름으로 바꿉니다.

```
aws rekognition compare-faces \
--source-image '{"S3Object":{"Bucket":"'bucket-name'", "Name":"source.jpg"} }' \
--target-image '{"S3Object":{"Bucket":"'bucket-name'", "Name":"target.jpg"} }'
```

Python

이 예제는 로컬 파일 시스템에서 불러오는 소스 이미지와 대상 이미지의 일치 얼굴에 대한 정보를 표시합니다.

sourceFile 값과 targetFile 값을 소스 및 대상 이미지의 파일 이름과 경로로 바꿉니다.

```
#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__":  
  
    sourceFile='source.jpg'  
    targetFile='target.jpg'  
    client=boto3.client('rekognition')  
  
    imageSource=open(sourceFile,'rb')  
    imageTarget=open(targetFile,'rb')  
  
    response=client.compare_faces(SimilarityThreshold=70,  
                                  SourceImage={'Bytes': imageSource.read()},  
                                  TargetImage={'Bytes': imageTarget.read()})  
  
    for faceMatch in response['FaceMatches']:  
        position = faceMatch['Face']['BoundingBox']  
        confidence = str(faceMatch['Face']['Confidence'])  
        print('The face at ' +  
              str(position['Left']) + ' ' +  
              str(position['Top']) +  
              ' matches with ' + confidence + '% confidence')  
  
    imageSource.close()  
    imageTarget.close()
```

.NET

이 예제는 로컬 파일 시스템에서 불러오는 소스 이미지와 대상 이미지의 일치 얼굴에 대한 정보를 표시합니다.

sourceImage 값과 targetImage 값을 소스 및 대상 이미지의 파일 이름과 경로로 바꿉니다.

```
//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 " + face.Confidence
        + "% confidence.");
}

Console.WriteLine("There was " + compareFacesResponse.UnmatchedFaces.Count
+ " face(s) that did not match");
Console.WriteLine("Source image rotation: " +
compareFacesResponse.SourceImageOrientationCorrection);
Console.WriteLine("Target image rotation: " +
compareFacesResponse.TargetImageOrientationCorrection);
}
}
```

CompareFaces 작업 요청

CompareFaces에 대한 입력은 이미지입니다. 이 예제에서는 소스 이미지와 대상 이미지를 로컬 파일 시스템에서 불러옵니다. SimilarityThreshold 입력 파라미터는 비교 얼굴을 응답에 포함시키는 기준이 되는 최소 신뢰도를 지정합니다. 자세한 내용은 [이미지 작업 \(p. 35\)](#) 단원을 참조하십시오.

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

CompareFaces 작업 응답

응답에서 얼굴 일치 배열, 원본 얼굴 정보, 원본 및 대상 이미지 방향, 일치하지 않는 얼굴 배열을 받습니다. 대상 이미지의 각 일치 얼굴에 대해 응답은 유사성 점수(소스 얼굴과 유사한 정도)와 얼굴 메타데이터를 제공합니다. 얼굴 메타데이터에는 일치 얼굴의 경계 상자와 얼굴 표식의 배열 등과 같은 정보가 포함됩니다. 일치하지 않는 얼굴의 배열에는 얼굴 메타데이터가 포함됩니다.

다음 응답 예에서는 다음에 유의하십시오.

- 얼굴 일치 정보 – 예제는 대상 이미지에서 찾은 하나의 얼굴 일치를 보여줍니다. 이 얼굴 일치의 경우, 경계 상자와 신뢰도 값(경계 상자에 얼굴이 포함될 Amazon Rekognition의 신뢰도 수준)이 제공됩니다. 99.99의 similarity 점수는 얼굴들이 얼마나 유사한지를 나타냅니다. 얼굴 일치 정보에는 표식 위치 배열도 포함됩니다.

여러 얼굴이 일치할 경우 faceMatches 배열에 모든 얼굴 일치가 포함됩니다.

- 원본 얼굴 정보 – 응답에는 경계 상자와 신뢰도 값을 포함해 비교에 사용했던 원본 이미지의 얼굴에 대한 정보가 포함됩니다.
- 이미지 방향 – 원본 및 대상 이미지의 방향에 관한 정보가 응답에 포함됩니다. Amazon Rekognition이 대상 이미지에서 일치하는 얼굴의 정확한 위치를 검색하고 이미지를 표시하려면 이것이 필요합니다. 자세한 내용은 [이미지 방향 및 경계 상자 좌표 가져오기 \(p. 47\)](#) 단원을 참조하십시오.
- 일치하지 않는 얼굴 일치 정보 – 예제는 Amazon Rekognition이 대상 이미지에서 발견한 얼굴로서, 원본 이미지에서 분석한 얼굴과 일치하지 않는 얼굴을 보여줍니다. 이 얼굴은 경계 상자와 신뢰도 값을 제공하여 경계 상자에 얼굴이 포함될 Amazon Rekognition의 신뢰도 수준을 알려줍니다. 얼굴 정보에는 표식 위치 배열도 포함됩니다.

Amazon Rekognition이 일치하지 않는 여러 얼굴을 찾을 경우 UnmatchedFaces 배열에는 일치하지 않은 모든 얼굴이 포함됩니다.

```
{  
    "FaceMatches": [ {  
        "Face": {  
            "BoundingBox": {  
                "Width": 0.5521978139877319,  
                "Top": 0.1203877404332161,  
                "Left": 0.23626373708248138,  
                "Height": 0.3126954436302185  
            },  
            "Confidence": 99.98751068115234,  
            "Pose": {  
                "Yaw": -82.36799621582031,  
                "Roll": -62.13221740722656,  
                "Pitch": 14.111111111111112  
            }  
        }  
    }]  
}
```

```
        "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
}
}
```

저장된 비디오에서 얼굴 감지

Amazon Rekognition Video는 Amazon S3 버킷에 저장된 비디오에서 얼굴을 감지하여 다음과 같은 정보를 제공할 수 있습니다.

- 비디오에서 얼굴이 감지된 시간
- 얼굴이 감지된 시점에 비디오 프레임에서 해당 얼굴의 위치
- 왼쪽 눈의 위치와 같은 얼굴 표식

Amazon Rekognition Video의 저장된 비디오 속 얼굴 감지는 비동기 작업입니다. 비디오에서 얼굴을 감지하려면 [StartFaceDetection \(p. 322\)](#)을 호출합니다. Amazon Rekognition Video는 비디오 분석의 완료 상태를 Amazon Simple Notification Service(Amazon SNS) 주제에 게시합니다. 비디오 분석이 성공적으로 완료되면 [GetFaceDetection \(p. 268\)](#)을 호출하여 비디오 분석 결과를 가져올 수 있습니다. 비디오 분석 시작 및 결과 가져오기에 대한 자세한 내용은 [Amazon Rekognition Video 작업 불러오기 \(p. 59\)](#) 단원을 참조하십시오.

이 절차는 비디오 분석 요청의 완료 상태를 가져오기 위해 Amazon Simple Queue Service(Amazon SQS) 대기열을 사용하는 [Java 또는 Python으로 Amazon S3 버킷에 저장된 비디오 분석\(SDK\) \(p. 66\)](#)의 코드를 확장합니다.

Amazon S3 버킷에 저장된 비디오에서 얼굴을 감지하려면(SDK)

1. [Java 또는 Python으로 Amazon S3 버킷에 저장된 비디오 분석\(SDK\) \(p. 66\)](#)을 수행합니다.
2. 1단계에서 만든 클래스 `VideoDetect`에 다음 코드를 추가합니다.

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 StartFaces(String bucket, String video) throws Exception{
```

```
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 GetResultsFaces() 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);

}
```

2a. main 함수에서 다음 줄을

```
StartLabels(bucket,video);
```

다음으로 바꿉니다.

```
StartFaces(bucket,video);
```

2b. 다음 줄을

```
GetResultsLabels();
```

다음으로 바꿉니다.

```
GetResultsFaces();
```

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.)  
  
def GetResultsFaces(self, jobId):  
    maxResults = 10  
    paginationToken = ''  
    finished = False  
  
    while finished == False:  
        response = self.rek.get_face_detection(JobId=jobId,  
                                                MaxResults=maxResults,  
                                                NextToken=paginationToken)  
  
        print(response['VideoMetadata']['Codec'])  
        print(str(response['VideoMetadata']['DurationMillis']))  
        print(response['VideoMetadata']['Format'])  
        print(response['VideoMetadata']['FrameRate'])  
  
        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
```

2a. main 함수에서 다음 줄을

```
response = self.rek.start_label_detection(Video={'S3Object': {'Bucket':  
self.bucket, 'Name': self.video}},  
                                         NotificationChannel={'RoleArn':  
self.roleArn, 'SNSTopicArn': self.topicArn})
```

다음으로 바꿉니다.

```
response = self.rek.start_face_detection(Video={'S3Object':  
{'Bucket':self.bucket,'Name':self.video}},  
                                         NotificationChannel={'RoleArn':self.roleArn,  
'SNSTopicArn':self.topicArn})
```

2b. 다음 줄을

```
self.GetResultsLabels(rekMessage['JobId'])
```

다음으로 바꿉니다.

```
self.GetResultsFaces(rekMessage['JobId'])
```

Note

Java 또는 Python으로 Amazon S3 버킷에 저장된 비디오 분석(SDK) (p. 66) 이외에 비디오 예제를 이미 실행한 경우, 바꿀 함수 이름이 다릅니다.

3. 코드를 실행합니다. 비디오에서 감지된 얼굴에 관한 정보가 표시됩니다.

GetFaceDetection 작업 응답

GetFaceDetection은 비디오에서 감지된 얼굴에 대한 정보가 포함된 배열(Faces)을 반환합니다. 배열 요소 FaceDetection (p. 362)은 비디오에서 얼굴이 감지될 때마다 존재합니다. 배열 요소는 비디오 시작 후 시 간별로(밀리초) 정렬되어 반환됩니다.

다음 예제는 GetFaceDetection의 부분 JSON 응답입니다. 응답에서 다음에 유의하십시오.

- 얼굴 정보 – FaceDetection 배열 요소에는 감지된 얼굴(FaceDetail (p. 359))과 비디오에서 얼굴이 감지된 시간(Timestamp)에 관한 정보가 담겨 있습니다.
- 페이지 정보 – 예제는 얼굴 감지 정보의 페이지 하나를 보여줍니다. GetFaceDetection의 MaxResults 입력 파라미터에 반환될 사람 요소의 수를 지정할 수 있습니다. MaxResults 보다 많은 결과가 존재할 경우 GetFaceDetection은 결과의 다음 페이지를 가져올 때 사용되는 토큰(NextToken)을 반환합니다. 자세한 내용은 Amazon Rekognition Video 분석 결과 가져오기 (p. 61) 단원을 참조하십시오.
- 비디오 정보 – GetFaceDetection이 반환하는 정보의 각 페이지에 있는 비디오 형식(VideoMetadata)에 관한 정가 응답에 포함됩니다.

```
{
    "Faces": [
        {
            "Face": {
                "BoundingBox": {
                    "Height": 0.2300000417232513,
                    "Left": 0.42500001192092896,
                    "Top": 0.16333332657814026,
                    "Width": 0.12937499582767487
                },
                "Confidence": 99.97504425048828,
                "Landmarks": [
                    {
                        "Type": "eyeLeft",
                        "X": 0.46415066719055176,
                        "Y": 0.2572723925113678
                    },
                    {
                        "Type": "eyeRight",
                        "X": 0.5068183541297913,
                        "Y": 0.23705792427062988
                    },
                    {
                        "Type": "nose",
                        "X": 0.49765899777412415,
                        "Y": 0.28383663296699524
                    },
                    {
                        "Type": "mouthLeft",
                        "X": 0.487221896648407,
                        "Y": 0.3452930748462677
                    }
                ]
            }
        }
    ]
}
```

```
        },
        {
            "Type": "mouthRight",
            "X": 0.5142884850502014,
            "Y": 0.33167609572410583
        }
    ],
    "Pose": {
        "Pitch": 15.966927528381348,
        "Roll": -15.547388076782227,
        "Yaw": 11.34195613861084
    },
    "Quality": {
        "Brightness": 44.80223083496094,
        "Sharpness": 99.95819854736328
    }
},
"Timestamp": 0
},
{
    "Face": {
        "BoundingBox": {
            "Height": 0.2000000298023224,
            "Left": 0.02999999329447746,
            "Top": 0.2199999988079071,
            "Width": 0.11249999701976776
        },
        "Confidence": 99.85971069335938,
        "Landmarks": [
            {
                "Type": "eyeLeft",
                "X": 0.06842322647571564,
                "Y": 0.3010137975215912
            },
            {
                "Type": "eyeRight",
                "X": 0.10543643683195114,
                "Y": 0.29697132110595703
            },
            {
                "Type": "nose",
                "X": 0.09569807350635529,
                "Y": 0.33701086044311523
            },
            {
                "Type": "mouthLeft",
                "X": 0.0732642263174057,
                "Y": 0.3757539987564087
            },
            {
                "Type": "mouthRight",
                "X": 0.10589495301246643,
                "Y": 0.3722417950630188
            }
        ],
        "Pose": {
            "Pitch": -0.5589138865470886,
            "Roll": -5.1093974113464355,
            "Yaw": 18.69594955444336
        },
        "Quality": {
            "Brightness": 43.052337646484375,
            "Sharpness": 99.68138885498047
        }
    },
    "Timestamp": 0
}
```

```
        },
        {
            "Face": {
                "BoundingBox": {
                    "Height": 0.2177777737379074,
                    "Left": 0.7593749761581421,
                    "Top": 0.13333334028720856,
                    "Width": 0.12250000238418579
                },
                "Confidence": 99.63436889648438,
                "Landmarks": [
                    {
                        "Type": "eyeLeft",
                        "X": 0.8005779385566711,
                        "Y": 0.20915353298187256
                    },
                    {
                        "Type": "eyeRight",
                        "X": 0.8391435146331787,
                        "Y": 0.21049551665782928
                    },
                    {
                        "Type": "nose",
                        "X": 0.8191410899162292,
                        "Y": 0.2523227035999298
                    },
                    {
                        "Type": "mouthLeft",
                        "X": 0.8093273043632507,
                        "Y": 0.29053622484207153
                    },
                    {
                        "Type": "mouthRight",
                        "X": 0.8366993069648743,
                        "Y": 0.29101791977882385
                    }
                ],
                "Pose": {
                    "Pitch": 3.165884017944336,
                    "Roll": 1.4182015657424927,
                    "Yaw": -11.151537895202637
                },
                "Quality": {
                    "Brightness": 28.910892486572266,
                    "Sharpness": 97.61507415771484
                }
            },
            "Timestamp": 0
        }.....
    ],
    "JobStatus": "SUCCEEDED",
    "NextToken": "i7fj5XPV/",
    "VideoMetadata": {
        "Codec": "h264",
        "DurationMillis": 67301,
        "FileExtension": "mp4",
        "Format": "QuickTime / MOV",
        "FrameHeight": 1080,
        "FrameRate": 29.970029830932617,
        "FrameWidth": 1920
    }
}
```

모음에서 얼굴 검색

Amazon Rekognition은 모음으로 알려진 서버 측 컨테이너에 감지된 얼굴에 대한 정보를 저장할 수 있습니다. 그런 다음 모음에 저장된 얼굴 정보를 사용하여 이미지, 저장된 비디오, 스트리밍 비디오에서 알려진 얼굴을 검색할 수 있습니다. Amazon Rekognition은 [IndexFaces \(p. 287\)](#) 작업을 지원합니다. 이 작업을 사용하여 이미지에서 얼굴을 감지하고, 감지된 얼굴 특징에 대한 정보를 모음으로 유지할 수 있습니다. 이것이 스토리지 기반 API 작업의 예인 이유는 서비스가 서버에서 정보를 유지하기 때문입니다.

얼굴 정보를 저장하려면 먼저 계정의 AWS 리전 중 하나에서 ([CreateCollection \(p. 224\)](#)) 얼굴 모음을 만들어야 합니다. [IndexFaces](#) 작업을 호출할 때 이 얼굴 모음을 지정합니다. 얼굴 모음을 만들고 모든 얼굴의 얼굴 특징 정보를 저장한 후에 모음에서 얼굴 일치를 검색할 수 있습니다. 이미지에서 얼굴을 검색하려면, [SearchFacesByImage \(p. 311\)](#)를 호출합니다. 저장된 비디오에서 얼굴을 검색하려면, [StartFaceSearch \(p. 326\)](#)를 호출합니다. 스트리밍 비디오에서 얼굴을 검색하려면, [CreateStreamProcessor \(p. 227\)](#)를 호출합니다.

Note

이 서비스는 실제 이미지 바이트를 유지하지 않습니다. 그 대신 기본 감지 알고리즘이 먼저 입력 이미지에서 얼굴을 감지하고 얼굴 특징을 각 얼굴의 특징 벡터로 추출한 다음 모음에 저장합니다. Amazon Rekognition은 얼굴 일치를 수행할 때 이러한 특징 벡터를 사용합니다.

여러 가지 일반적인 사용자 시나리오를 위한 자습서 모음. 예를 들어 [IndexFaces](#) 작업을 사용하여 얼굴 모음을 만들어 스캔한 배지 이미지를 저장할 수 있습니다. 직원이 건물에 들어오면 직원 얼굴 이미지가 캡처되어 [SearchFacesByImage](#) 작업으로 전송됩니다. 얼굴 일치에서 충분히 높은 유사성 점수(가령 99%)가 나으면 직원을 인증할 수 있습니다.

모음 관리

얼굴 모음은 기본 Amazon Rekognition 리소스이며, 생성되는 각각의 얼굴 모음에는 고유의 Amazon Resource Name(ARN)이 있습니다. 계정의 특정 AWS 리전에서 각각의 얼굴 모음을 만듭니다. 모음을 만들면, 최신 버전의 얼굴 감지 모델에 연결됩니다. 자세한 내용은 [모델 버전 관리 \(p. 9\)](#) 단원을 참조하십시오.

모음에 대해 다음 관리 작업을 수행할 수 있습니다.

- the section called “[CreateCollection](#)” (p. 224)을 사용하여 모음을 만듭니다. 자세한 내용은 [모음 만들기 \(p. 128\)](#) 단원을 참조하십시오.
- the section called “[ListCollections](#)” (p. 295)를 사용하여 사용 가능 모음을 조회합니다. 자세한 내용은 [모음 나열 \(p. 131\)](#) 단원을 참조하십시오.
- the section called “[DeleteCollection](#)” (p. 230)을 사용하여 모음을 삭제합니다. 자세한 내용은 [모음 삭제 \(p. 134\)](#) 단원을 참조하십시오.

모음에서 얼굴 관리

얼굴 모음을 만든 후 모음에 얼굴을 저장할 수 있습니다. Amazon Rekognition은 모음의 얼굴을 관리하기 위해 다음 작업을 제공합니다.

- the section called “[IndexFaces](#)” (p. 287) 작업은 입력 이미지(JPEG 또는 PNG)에서 얼굴을 감지하여 저장된 얼굴 모음에 추가합니다. 이미지에서 감지된 각 얼굴에 대해 고유한 얼굴 ID가 반환됩니다. 얼굴을 유지한 후 얼굴 모음에서 얼굴 일치를 검색할 수 있습니다. 자세한 내용은 [모음에 얼굴 추가 \(p. 137\)](#) 단원을 참조하십시오.
- the section called “[ListFaces](#)” (p. 298) 작업은 모음의 얼굴을 나열합니다. 자세한 내용은 [모음에 얼굴 추가 \(p. 137\)](#) 단원을 참조하십시오.

- the section called “DeleteFaces” (p. 232) 작업은 모음에서 얼굴을 삭제합니다. 자세한 내용은 모음에서 얼굴 삭제 (p. 149) 단원을 참조하십시오.

모음에 있는 얼굴 검색

얼굴 모음을 만들고 얼굴을 저장한 후 얼굴 모음에서 얼굴 일치를 검색할 수 있습니다. Amazon Rekognition에서는 다음과 일치하는 얼굴을 모음에서 검색할 수 있습니다.

- 제공된 얼굴 ID(the section called “SearchFaces” (p. 308)). 자세한 내용은 얼굴 ID를 사용하여 얼굴 검색 (p. 152) 단원을 참조하십시오.
- 제공된 이미지에서 가장 큰 얼굴(the section called “SearchFacesByImage” (p. 311)). 자세한 내용은 이미지를 사용하여 얼굴 검색 (p. 156) 단원을 참조하십시오.
- 저장된 비디오의 얼굴. 자세한 내용은 저장된 비디오에서 얼굴 검색 (p. 160) 단원을 참조하십시오.
- 스트리밍 비디오의 얼굴. 자세한 내용은 스트리밍 비디오 작업 (p. 84) 단원을 참조하십시오.

CompareFaces 작업과 얼굴 검색 작업은 다음과 같은 차이가 있습니다.

- CompareFaces 작업은 원본 이미지의 얼굴과 대상 이미지의 얼굴을 비교합니다. 이 비교의 범위는 대상 이미지에서 감지된 얼굴로 제한됩니다. 자세한 내용은 이미지에 있는 얼굴 비교 (p. 114) 단원을 참조하십시오.
- SearchFaces와 SearchFacesByImage는 (FaceId 또는 입력 이미지에 의해 식별된) 얼굴을 주어진 얼굴 모음의 모든 얼굴과 비교합니다. 따라서 이 검색의 범위가 훨씬 큽니다. 또한 얼굴 모음에 이미 저장된 얼굴들은 얼굴 특징 정보가 유지되기 때문에 일치하는 얼굴을 여러 번 검색할 수 있습니다.

Note

유사성 임계값을 입력 파라미터로 제공하여 모든 검색 작업(CompareFaces (p. 219), SearchFaces (p. 308), SearchFacesByImage (p. 311))의 결과를 제어할 수 있습니다.

유사성 임계값(FaceMatchThreshold)은 일치하는 얼굴의 유사성에 따라 반환되는 결과 수를 제어합니다. 이 임계값은 일치 결과에 포함된 가양성의 수를 결정할 수 있기 때문에 사용 사례에 맞게 보정하기 위해 중요합니다. 이 값은 검색 결과의 회수를 제어합니다. 임계값이 낮을수록 회수가 많아집니다.

모든 머신 러닝 시스템은 확률적이므로 사용 사례에 따라 자체 판단을 근거로 올바른 유사성 임계값을 설정해야 합니다. 예를 들어 비슷하게 생긴 가족을 식별하는 포토 앱을 만들려는 경우 임계값을 낮게(예:80%) 선택할 수 있습니다. 반면, 많은 법 집행 사용 사례의 경우, 실수로 잘못 식별되는 경우를 줄이기 위해 99% 이상의 높은 임계값을 사용하는 것이 좋습니다.

FaceMatchThreshold 외에, 실수로 잘못 식별되는 경우를 줄이기 위한 수단으로 Similarity 응답 속성을 사용할 수 있습니다. 예를 들어 낮은 임계값(예: 80%)을 사용하여 보다 많은 결과를 반환하게 한 후 응답 속성 Similarity(유사성의 정도 %)를 사용하여 선택 범위를 좁히고 해당 애플리케이션에서 올바른 응답을 필터링할 수 있습니다. 다시 유사성을 높이면(예: 99% 이상) 잘못된 식별이 줄어들 것입니다.

모음 만들기

CreateCollection (p. 224) 작업을 사용하여 모음을 만들 수 있습니다.

자세한 내용은 모음 관리 (p. 127) 단원을 참조하십시오.

모음을 만들려면(SDK)

- 아직 설정하지 않았다면 다음과 같이 하십시오.

- a. IAM 사용자를 만들거나 업데이트하여 AmazonRekognitionFullAccess 권한을 부여합니다. 자세한 내용은 [1단계: AWS 계정 설정 및 IAM 사용자 만들기 \(p. 11\)](#) 단원을 참조하십시오.
- b. AWS CLI와 AWS SDK를 설치하고 구성합니다. 자세한 내용은 [2단계: AWS CLI 및 AWS SDK 설정 \(p. 11\)](#) 단원을 참조하십시오.

2. 다음 예제를 사용하여 CreateCollection 작업을 호출합니다.

Java

다음 예제는 모음을 만들고 Amazon Resource Name(ARN)을 표시합니다.

collectionId의 값을, 만들려는 모음의 이름으로 변경합니다.

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

AWS CLI

이 AWS CLI 명령은 create-collection CLI 작업에 대한 JSON 출력을 표시합니다.

collection-id의 값을, 만들려는 모음의 이름으로 바꿉니다.

```
aws rekognition create-collection \
```

```
--collection-id "collectionname"
```

Python

다음 예제는 모음을 만들고 Amazon Resource Name(ARN)을 표시합니다.

collectionId의 값을, 만들려는 모음의 이름으로 변경합니다.

```
#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__":  
  
    maxResults=2  
    collectionId='MyCollection'  
  
    client=boto3.client('rekognition')  
  
    #Create a collection  
    print('Creating collection:' + collectionId)  
    response=client.create_collection(CollectionId=collectionId)  
    print('Collection ARN: ' + response['CollectionArn'])  
    print('Status code: ' + str(response['StatusCode']))  
    print('Done...')
```

.NET

다음 예제는 모음을 만들고 Amazon Resource Name(ARN)을 표시합니다.

collectionId의 값을, 만들려는 모음의 이름으로 변경합니다.

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

CreateCollection 작업 요청

creationCollection에 대한 입력은 생성하려는 모음의 이름입니다.

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

CreateCollection 작업 응답

Amazon Rekognition은 모음을 만들고 새로 만든 모음의 Amazon Resource Name(ARN)을 반환합니다.

```
{
    "CollectionArn": "aws:rekognition:us-east-1:acct-id:collection/examplecollection",
    "StatusCode": 200
}
```

모음 나열

[ListCollections \(p. 295\)](#) 작업을 사용하여 현재 사용하는 리전의 모음을 나열할 수 있습니다.

자세한 내용은 [모음 관리 \(p. 127\)](#) 단원을 참조하십시오.

모음을 나열하려면(SDK)

1. 아직 설정하지 않았다면 다음과 같이 하십시오.
 - a. IAM 사용자를 만들거나 업데이트하여 AmazonRekognitionFullAccess 권한을 부여합니다. 자세한 내용은 [1단계: AWS 계정 설정 및 IAM 사용자 만들기 \(p. 11\)](#) 단원을 참조하십시오.
 - b. AWS CLI와 AWS SDK를 설치하고 구성합니다. 자세한 내용은 [2단계: AWS CLI 및 AWS SDK 설정 \(p. 11\)](#) 단원을 참조하십시오.
2. 다음 예제를 사용하여 ListCollections 작업을 호출합니다.

Java

다음 예제는 현재 리전의 모음을 나열합니다.

```
//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 {

    RotateImageAmazonRekognition 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);

}
}
```

AWS CLI

O| AWS CLI 명령은 list-collections CLI 작업에 대한 JSON 출력을 표시합니다.

```
aws rekognition list-collections
```

Python

다음 예제는 현재 리전의 모음을 나열합니다.

```
#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__":
    maxResults=2

    client=boto3.client('rekognition')

    #Display all the collections
    print('Displaying collections...')
    response=client.list_collections(MaxResults=maxResults)

    while True:
        collections=response['CollectionIds']
```

```
for collection in collections:  
    print (collection)  
    if 'NextToken' in response:  
        nextToken=response['NextToken']  
  
response=client.list_collections(NextToken=nextToken,MaxResults=maxResults)  
  
    else:  
        break  
  
    print('done...')
```

.NET

다음 예제는 현재 리전의 모음을 나열합니다.

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

ListCollections 작업 요청

ListCollections에 대한 입력은 반환될 최대 모음 수입니다.

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

응답의 얼굴이, `MaxResults`가 요청한 것보다 많으면 후속 `ListCollections` 호출에서 다음 결과 집합을 가져오는 데 사용할 수 있는 토큰이 반환됩니다. 다음 예를 참조하십시오.

```
{  
    "NextToken": "MGYZLAHX1T5a....",  
    "MaxResults": 2  
}
```

ListCollections 작업 응답

Amazon Rekognition은 모음 배열(`CollectionIds`)을 반환합니다. 각 모음의 얼굴을 분석하는 데 사용된 얼굴 모델 버전은 별도의 배열(`FaceModelVersions`)로 제공됩니다. 예를 들어 다음 JSON 응답에서 `MyCollection` 모음은 얼굴 모델 버전 2.0을 사용하여 얼굴을 분석합니다. `AnotherCollection` 모음은 얼굴 모델 버전 3.0을 사용합니다. 자세한 내용은 [모델 버전 관리 \(p. 9\)](#) 단원을 참조하십시오.

`NextToken`은 `ListCollections`에 대한 후속 호출에서 다음 결과 집합을 가져오는 데 사용되는 토큰입니다.

```
{  
    "CollectionIds": [  
        "MyCollection",  
        "AnotherCollection"  
    ],  
    "FaceModelVersions": [  
        "2.0",  
        "3.0"  
    ],  
    "NextToken": "MGYZLAHX1T5a...."  
}
```

모음 삭제

[DeleteCollection \(p. 230\)](#) 작업을 사용하여 모음을 삭제할 수 있습니다.

자세한 내용은 [모음 관리 \(p. 127\)](#) 단원을 참조하십시오.

모음을 삭제하려면(SDK)

1. 아직 설정하지 않았다면 다음과 같이 하십시오.
 - a. IAM 사용자를 만들거나 업데이트하여 `AmazonRekognitionFullAccess` 권한을 부여합니다. 자세한 내용은 [1단계: AWS 계정 설정 및 IAM 사용자 만들기 \(p. 11\)](#) 단원을 참조하십시오.
 - b. AWS CLI와 AWS SDK를 설치하고 구성합니다. 자세한 내용은 [2단계: AWS CLI 및 AWS SDK 설정 \(p. 11\)](#) 단원을 참조하십시오.
2. 다음 예제를 사용하여 `DeleteCollection` 작업을 호출합니다.

Java

이 예제는 모음을 삭제합니다.

`collectionId` 값을, 삭제하려는 모음으로 변경합니다.

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

AWS CLI

이 AWS CLI 명령은 `delete-collection` CLI 작업에 대한 JSON 출력을 표시합니다.
`collection-id`의 값을, 삭제하려는 모음의 이름으로 바꿉니다.

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

Python

이 예제는 모음을 삭제합니다.

`collectionId` 값을, 삭제하려는 모음으로 변경합니다.

```
#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
```

```
if __name__ == "__main__":
    collectionId='MyCollection'
    print('Attempting to delete collection ' + collectionId)
    client=boto3.client('rekognition')
    statusCode=''
    try:
        response=client.delete_collection(CollectionId=collectionId)
        statusCode=response['StatusCode']

    except ClientError as e:
        if e.response['Error']['Code'] == 'ResourceNotFoundException':
            print ('The collection ' + collectionId + ' was not found ')
        else:
            print ('Error other than Not Found occurred: ' + e.response['Error']
['Message'])
        statusCode=e.response['ResponseMetadata']['HTTPStatusCode']
        print('Operation returned Status Code: ' + str(statusCode))
    print('Done...')
```

.NET

이 예제는 모음을 삭제합니다.

collectionId 값을, 삭제하려는 모음으로 변경합니다.

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

DeleteCollection 작업 요청

DeleteCollection에 대한 입력은 삭제할 모음의 ID입니다. 다음 JSON 예제를 참조하십시오.

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

DeleteCollection 작업 응답

DeleteCollection 응답에는 작업의 성공 또는 실패를 나타내는 HTTP 상태 코드가 들어 있습니다. 모음이 성공적으로 삭제되면 200이 반환됩니다.

```
{"StatusCode":200}
```

모음에 얼굴 추가

[IndexFaces \(p. 287\)](#) 작업을 사용하여 이미지에서 얼굴을 감지하고 모음에 추가할 수 있습니다. Amazon Rekognition은 감지된 각 얼굴에서 얼굴 특징을 추출하여 데이터베이스에 이 특징 정보를 저장합니다. 또한 이 명령은 감지된 각 얼굴의 메타데이터를 지정된 얼굴 모음에 저장합니다. Amazon Rekognition은 실제 이미지 바이트를 저장하지 않습니다.

IndexFaces 작업은 각 얼굴에 대해 다음의 정보를 유지합니다.

- **다차원 얼굴 특징** – IndexFaces는 얼굴 분석을 사용하여 얼굴 특징에 대한 다차원 정보를 추출하고, 이 정보를 얼굴 모음에 저장합니다. 이 정보를 직접 액세스할 수는 없습니다. 이 정보는 Amazon Rekognition이 얼굴 모음에서 얼굴 일치를 검색할 때 사용합니다.
- **메타데이터** – 각 얼굴의 메타데이터에는 경계 상자, 신뢰도 수준(경계 상자에 얼굴이 포함될 신뢰도 수준), Amazon Rekognition이 할당한 ID(얼굴 ID 및 이미지 ID), 요청 내 외부 이미지 ID(제공한 경우)가 포함됩니다. 이 정보는 IndexFaces API 호출에 대한 응답으로 반환됩니다. 예를 들어 다음 응답 예의 face 요소를 참조하십시오.

이 서비스는 다음 API 호출에 대한 응답으로 이 메타데이터를 반환합니다.

- [the section called “ListFaces” \(p. 298\)](#)
- **얼굴 검색 작업** – [the section called “SearchFaces” \(p. 308\)](#) 및 [the section called “SearchFacesByImage” \(p. 311\)](#)에 대한 응답은 일치하는 얼굴의 일치 신뢰도와 함께 일치한 얼굴의 이 메타데이터를 반환합니다.

IndexFaces에서 인덱스를 생성하는 얼굴 수는 입력 모음과 연결된 얼굴 감지 모델의 버전에 따라 달립니다. 자세한 내용은 [모델 버전 관리 \(p. 9\)](#) 단원을 참조하십시오.

감지된 이미지와 인덱싱된 얼굴을 연결해야 합니다. 예를 들어 이미지와 이미지의 얼굴에서 클라이언트 측 인덱스를 유지해야 합니다. 얼굴을 이미지와 연결하려면, ExternalImageId 요청 파라미터에서 이미지 ID를 지정합니다. 이미지 ID는 만드는 파일 이름이나 다른 ID일 수 있습니다.

이 API는 얼굴 모음에서 유지하는 위의 정보 외에, 모음에 유지되지 않는 얼굴 세부 정보도 반환합니다. 다음 예제 응답에서 faceDetail 요소를 참조하십시오.

Note

`DetectFaces`는 동일한 정보를 반환하므로 같은 이미지에 대해 `DetectFaces`와 `IndexFaces`를 모두 호출할 필요는 없습니다.

자세한 내용은 [모음에서 얼굴 관리 \(p. 127\)](#) 단원을 참조하십시오.

모음에 얼굴을 추가하려면(SDK)

1. 아직 설정하지 않았다면 다음과 같이 하십시오.
 - a. `AmazonRekognitionFullAccess` 권한과 `AmazonS3ReadOnlyAccess` 권한을 가진 IAM 사용자를 만들어가 업데이트합니다. 자세한 내용은 [1단계: AWS 계정 설정 및 IAM 사용자 만들기 \(p. 11\)](#) 단원을 참조하십시오.
 - b. AWS CLI와 AWS SDK를 설치하고 구성합니다. 자세한 내용은 [2단계: AWS CLI 및 AWS SDK 설정 \(p. 11\)](#) 단원을 참조하십시오.
2. (하나 이상의 얼굴이 있는) 이미지를 Amazon S3 버킷에 업로드합니다.

이에 관한 지침은 Amazon Simple Storage Service 콘솔 사용 설명서의 [Amazon S3에 객체 업로드](#)를 참조하십시오.

3. 다음 예제를 사용하여 `IndexFaces` 작업을 호출합니다.

Java

이 예제는 모음에 추가된 얼굴의 얼굴 식별자를 표시합니다.

`collectionId`의 값을, 얼굴을 추가하려는 모음의 이름으로 변경합니다. `bucket` 및 `photo`의 값을 2단계에서 사용한 Amazon S3 버킷과 이미지의 이름으로 바꿉니다.

```
//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.S3Object;
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)
    .withCollectionId(collectionId)
    .withExternalImageId(photo)
    .withDetectionAttributes("ALL");

IndexFacesResult
indexFacesResult=rekognitionClient.indexFaces(indexFacesRequest);

System.out.println(photo + " added");
List < FaceRecord > faceRecords = indexFacesResult.getFaceRecords();
for (FaceRecord faceRecord: faceRecords) {
    System.out.println("Face detected: Faceid is " +
        faceRecord.getFace().getFaceId());
}
}
```

AWS CLI

이 AWS CLI 명령은 `index-faces` CLI 작업에 대한 JSON 출력을 표시합니다.

`collection-id`의 값을, 얼굴을 저장할 모음 이름으로 바꿉니다. `Bucket` 및 `Name`의 값을, 2단계에서 사용한 Amazon S3 버킷과 이미지 파일로 바꿉니다.

```
aws rekognition index-faces \
--image '{"S3Object":{"Bucket":"bucket-name","Name":"file-name"}}' \
--collection-id "collection-id" \
--detection-attributes "ALL" \
--external-image-id "example-image.jpg"
```

Python

이 예제는 모음에 추가된 얼굴의 얼굴 식별자를 표시합니다.

`collectionId`의 값을, 얼굴을 추가하려는 모음의 이름으로 변경합니다. `bucket` 및 `photo`의 값을 2단계에서 사용한 Amazon S3 버킷과 이미지의 이름으로 바꿉니다.

```
#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'

    client=boto3.client('rekognition')

    response=client.index_faces

    response=client.index_faces(CollectionId=collectionId,
                                Image={'S3Object':{'Bucket':bucket,'Name':image}},
```

```
ExternalImageId=fileName,
DetectionAttributes=['ALL'])

print ('Faces in ' + fileName)
for faceRecord in response['FaceRecords']:
    print (faceRecord['Face']['FaceId'])
```

.NET

이 예제는 모음에 추가된 얼굴의 얼굴 식별자를 표시합니다.

collectionId의 값을, 얼굴을 추가하려는 모음의 이름으로 변경합니다. bucket 및 photo의 값을 2단계에서 사용한 Amazon S3 버킷과 이미지의 이름으로 바꿉니다.

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

IndexFaces 작업 요청

IndexFaces에 대한 입력은 인덱스를 생성할 이미지와, 얼굴을 추가할 모음입니다.

```
{  
    "CollectionId": "MyCollection",  
    "Image": {  
        "S3Object": {  
            "Bucket": "bucket",  
            "Name": "input.jpg"  
        }  
    },  
    "ExternalImageId": "input.jpg",  
    "DetectionAttributes": [  
        "ALL"  
    ]  
}
```

IndexFaces 작업 응답

IndexFaces는 이미지에서 감지된 얼굴에 대한 정보를 반환합니다. 예를 들어 다음 JSON 응답에는 입력 이미지에서 감지된 얼굴의 기본 감지 속성이 포함됩니다.

```
{  
    "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": {  
                "BoundingBox": {  
                    "Height": 0.06333333253860474,  
                    "Left": 0.17185185849666595,  
                    "Top": 0.7366666793823242,  
                    "Width": 0.11061728745698929  
                },  
                "Confidence": 99.99999237060547,  
                "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
        }
    ],
    "Pose": {
        "Pitch": -0.9409101605415344,
        "Roll": 7.233824253082275,
        "Yaw": -2.3602254390716553
    },
    "Quality": {
        "Brightness": 32.01998519897461,
        "Sharpness": 93.67259216308594
    }
}
},
],
"OrientationCorrection": "ROTATE_0"
}
```

모든 얼굴 정보를 가져오려면 `DetectionAttributes` 요청 파라미터에서 'ALL'을 지정합니다. 예를 들어 다음 예제 응답에서 `faceDetail` 요소의 추가 정보에 유의해야 하며 이것은 서버에서 지속되지 않습니다.

- 25개의 얼굴 표식(앞의 예의 단 5개와 비교)
- 얼굴 속성 9개(안경, 턱수염 등)
- 감정(emotion 요소 참조)

`face` 요소는 서버에서 유지되는 메타데이터를 제공합니다.

`FaceModelVersion`은 모음과 연결된 얼굴 모델의 버전입니다. 자세한 내용은 [모델 버전 관리 \(p. 9\)](#) 단원을 참조하십시오.

`OrientationCorrection`은 이미지의 예상 방향입니다. 자세한 내용은 [이미지 방향 및 경계 상자 좌표 가져오기 \(p. 47\)](#) 단원을 참조하십시오.

```
{
    "FaceModelVersion": "3.0",
    "FaceRecords": [
        {
            "Face": {
                "BoundingBox": {
                    "Height": 0.0633333253860474,
                    "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": 0.0633333253860474,
        "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"
}
```

모음에 얼굴 나열

[ListFaces \(p. 298\)](#) 작업을 사용하여 모음에 얼굴을 나열할 수 있습니다.

자세한 내용은 [모음에서 얼굴 관리 \(p. 127\)](#) 단원을 참조하십시오.

모음에 있는 얼굴을 나열하려면(SDK)

1. 아직 설정하지 않았다면 다음과 같이 하십시오.
 - a. IAM 사용자를 만들거나 업데이트하여 AmazonRekognitionFullAccess 권한을 부여합니다. 자세한 내용은 [1단계: AWS 계정 설정 및 IAM 사용자 만들기 \(p. 11\)](#) 단원을 참조하십시오.
 - b. AWS CLI와 AWS SDK를 설치하고 구성합니다. 자세한 내용은 [2단계: AWS CLI 및 AWS SDK 설정 \(p. 11\)](#) 단원을 참조하십시오.
2. 다음 예제를 사용하여 `ListFaces` 작업을 호출합니다.

Java

이 예제는 모음의 얼굴 목록을 표시합니다.

`collectionId`의 값을 원하는 모음으로 변경합니다.

```
//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);
}

}
```

AWS CLI

이 AWS CLI 명령은 list-faces CLI 작업에 대한 JSON 출력을 표시합니다. collection-id의 값을, 목록을 조회할 모음의 이름으로 바꿉니다.

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

Python

이 예제는 모음의 얼굴 목록을 표시합니다.

collectionId의 값을 원하는 모음으로 변경합니다.

```
#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'
    maxResults=2
    tokens=True

    client=boto3.client('rekognition')
    response=client.list_faces(CollectionId=collectionId,
                                MaxResults=maxResults)

    print('Faces in collection ' + collectionId)

    while tokens:

        faces=response['Faces']

        for face in faces:
            print (face)
        if 'NextToken' in response:
            nextToken=response['NextToken']
            response=client.list_faces(CollectionId=collectionId,
                                        NextToken=nextToken,MaxResults=maxResults)
        else:
            tokens=False
```

.NET

이 예제는 모음의 얼굴 목록을 표시합니다.

collectionId의 값을 원하는 모음으로 변경합니다.

```
//Copyright 2018 Amazon.com, Inc. or its affiliates. All Rights Reserved.  
//PDX-License-Identifier: MIT-0 (For details, see https://github.com/awsdocs/  
amazon-rekognition-developer-guide/blob/master/LICENSE-SAMPLECODE.)  
  
using System;  
using Amazon.Rekognition;  
using Amazon.Rekognition.Model;  
  
public class ListFaces  
{  
    public static void Example()  
    {  
        String collectionId = "MyCollection";  
  
        AmazonRekognitionClient rekognitionClient = new AmazonRekognitionClient();  
  
        ListFacesResponse listFacesResponse = null;  
        Console.WriteLine("Faces in collection " + collectionId);  
  
        String paginationToken = null;  
        do  
        {  
            if (listFacesResponse != null)  
                paginationToken = listFacesResponse.NextToken;  
  
            ListFacesRequest listFacesRequest = new ListFacesRequest()  
            {  
                CollectionId = collectionId,  
                MaxResults = 1,  
                NextToken = paginationToken  
            };  
  
            listFacesResponse = rekognitionClient.ListFaces(listFacesRequest);  
            foreach(Face face in listFacesResponse.Faces)  
                Console.WriteLine(face.FaceId);  
        } while (listFacesResponse != null && !  
String.IsNullOrEmpty(listFacesResponse.NextToken));  
    }  
}
```

ListFaces 작업 요청

ListFaces에 대한 입력은 얼굴 목록을 조회하려는 모음의 ID입니다. MaxResults는 반환할 최대 얼굴 수입니다.

```
{  
    "CollectionId": "MyCollection",  
    "MaxResults": 1  
}
```

응답의 얼굴이 MaxResults가 요청한 것보다 많으면 후속 ListFaces 호출에서 다음 결과 집합을 가져오는데 사용할 수 있는 토큰이 반환됩니다. 예:

```
{  
    "CollectionId": "MyCollection",  
    "NextToken": "sm+5ythT3aeEVIR4WA....",  
    "MaxResults": 1
```

}

ListFaces 작업 응답

ListFaces의 응답은 지정한 모음에 저장된 얼굴 메타데이터에 대한 정보입니다.

- FaceModelVersion – 모음과 연결된 얼굴 모델의 버전입니다. 자세한 내용은 [모델 버전 관리 \(p. 9\)](#) 단원을 참조하십시오.
- Faces – 모음에 있는 얼굴에 대한 정보입니다. [the section called “BoundingBox” \(p. 344\)](#), 신뢰도, 이미지 식별자, 얼굴 ID 등에 대한 정보가 포함됩니다. 자세한 내용은 [the section called “Face” \(p. 357\)](#) 단원을 참조하십시오.
- NextToken – 다음 결과 집합을 가져오는 데 사용되는 토큰입니다.

```
{  
    "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"  
        }  
    ],  
    "NextToken": "sm+5ythT3aeEVIR4WA...."  
}
```

모음에서 얼굴 삭제

[DeleteFaces \(p. 232\)](#) 작업을 사용하여 모음에서 얼굴을 삭제할 수 있습니다. 자세한 내용은 [모음에서 얼굴 관리 \(p. 127\)](#) 단원을 참조하십시오.

모음에서 얼굴을 삭제하려면

1. 아직 설정하지 않았다면 다음과 같이 하십시오.
 - a. IAM 사용자를 만들거나 업데이트하여 AmazonRekognitionFullAccess 권한을 부여합니다. 자세한 내용은 [1단계: AWS 계정 설정 및 IAM 사용자 만들기 \(p. 11\)](#) 단원을 참조하십시오.
 - b. AWS CLI와 AWS SDK를 설치하고 구성합니다. 자세한 내용은 [2단계: AWS CLI 및 AWS SDK 설정 \(p. 11\)](#) 단원을 참조하십시오.
2. 다음 예제를 사용하여 DeleteCollection 작업을 호출합니다.

Java

이 예제는 모음에서 한 개의 얼굴을 삭제합니다.

collectionId의 값을, 삭제하려는 얼굴을 포함하는 모음으로 변경합니다. faces의 값을, 삭제하려는 얼굴 ID로 변경합니다. 여러 얼굴을 삭제하려면 faces 배열에 얼굴 ID를 추가합니다.

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

AWS CLI

이 AWS CLI 명령은 delete-collection CLI 작업에 대한 JSON 출력을 표시합니다.
collection-id의 값을, 삭제하려는 얼굴을 포함하는 모음의 이름으로 바꿉니다. face-ids의 값을, 삭제하려는 얼굴 ID 배열로 바꿉니다.

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

Python

이 예제는 모음에서 한 개의 얼굴을 삭제합니다.

collectionId의 값을, 삭제하려는 얼굴을 포함하는 모음으로 변경합니다. faces의 값을, 삭제하려는 얼굴 ID로 변경합니다. 여러 얼굴을 삭제하려면 faces 배열에 얼굴 ID를 추가합니다.

```
#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__":  
  
    collectionId='MyCollection'  
    faces=[]  
    faces.append("xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx")  
  
    client=boto3.client('rekognition')  
  
    response=client.delete_faces(CollectionId=collectionId,  
                                  FaceIds=faces)  
  
    print(str(len(response['DeletedFaces'])) + ' faces deleted:')  
    for faceId in response['DeletedFaces']:  
        print (faceId)
```

.NET

이 예제는 모음에서 한 개의 얼굴을 삭제합니다.

collectionId의 값을, 삭제하려는 얼굴을 포함하는 모음으로 변경합니다. faces의 값을, 삭제하려는 얼굴 ID로 변경합니다. 여러 얼굴을 삭제하려면 faces 목록에 얼굴 ID를 추가합니다.

```
//Copyright 2018 Amazon.com, Inc. or its affiliates. All Rights Reserved.  
//PDX-License-Identifier: MIT-0 (For details, see https://github.com/awsdocs/amazon-rekognition-developer-guide/blob/master/LICENSE-SAMPLECODE.)  
  
using System;  
using System.Collections.Generic;  
using Amazon.Rekognition;  
using Amazon.Rekognition.Model;  
  
public class DeleteFaces  
{  
    public static void Example()  
    {  
        String collectionId = "MyCollection";  
        List<String> faces = new List<String>() { "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" };  
  
        AmazonRekognitionClient rekognitionClient = new AmazonRekognitionClient();  
  
        DeleteFacesRequest deleteFacesRequest = new DeleteFacesRequest()  
        {  
            CollectionId = collectionId,  
            FaceIds = faces  
        };  
  
        DeleteFacesResponse deleteFacesResponse =  
rekognitionClient.DeleteFaces(deleteFacesRequest);  
        foreach (String face in deleteFacesResponse.DeletedFaces)  
            Console.WriteLine("FaceID: " + face);  
    }  
}
```

DeleteFaces 작업 요청

DeleteFaces에 대한 입력은 얼굴을 포함하는 모음의 ID와, 삭제할 얼굴의 얼굴 ID 배열입니다.

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

DeleteFaces 작업 응답

DeleteFaces 응답에는 삭제된 얼굴의 얼굴 ID 배열이 들어 있습니다.

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

얼굴 ID를 사용하여 얼굴 검색

SearchFaces (p. 308) 작업을 사용하여 제공된 얼굴 ID와 일치하는 모음에서 얼굴을 검색할 수 있습니다.

얼굴이 감지되고 모음에 추가되면 IndexFaces (p. 287) 작업 응답에서 얼굴 ID가 반환됩니다. 자세한 내용은 [모음에서 얼굴 관리 \(p. 127\)](#) 단원을 참조하십시오.

얼굴 ID(SDK)를 사용하여 모음에서 얼굴을 검색하려면

1. 아직 설정하지 않았다면 다음과 같이 하십시오.
 - a. IAM 사용자를 만들거나 업데이트하여 AmazonRekognitionFullAccess 권한을 부여합니다. 자세한 내용은 [1단계: AWS 계정 설정 및 IAM 사용자 만들기 \(p. 11\)](#) 단원을 참조하십시오.
 - b. AWS CLI와 AWS SDK를 설치하고 구성합니다. 자세한 내용은 [2단계: AWS CLI 및 AWS SDK 설정 \(p. 11\)](#) 단원을 참조하십시오.
2. 다음 예제를 사용하여 SearchFaces 작업을 호출합니다.

Java

이 예제는 ID로 식별한 얼굴과 일치하는 얼굴에 대한 정보를 표시합니다.

collectionID의 값을, 원하는 얼굴이 있는 모음으로 변경합니다. faceId의 값을, 찾으려는 얼굴의 식별자로 변경합니다.

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

예제 코드를 실행합니다. 일치하는 얼굴에 대한 정보가 표시됩니다.

AWS CLI

이 AWS CLI 명령은 `search-faces` CLI 작업에 대한 JSON 출력을 표시합니다. 검색하려는 얼굴 식별자로 `face-id` 값을 바꾸고, 검색하려는 모음으로 `collection-id`를 바꿉니다.

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

Python

이 예제는 ID로 식별한 얼굴과 일치하는 얼굴에 대한 정보를 표시합니다.

`collectionId`의 값을, 원하는 얼굴이 있는 모음으로 변경합니다. `faceId`의 값을, 찾으려는 얼굴의 식별자로 변경합니다.

```
#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'
threshold = 50
maxFaces=2
faceId='xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx'

client=boto3.client('rekognition')

response=client.search_faces(CollectionId=collectionId,
                               FaceId=faceId,
                               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

이 예제는 ID로 식별한 얼굴과 일치하는 얼굴에 대한 정보를 표시합니다.

collectionID의 값을, 원하는 얼굴이 있는 모음으로 변경합니다. faceId의 값을, 찾으려는 얼굴의 식별자로 변경합니다.

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

예제 코드를 실행합니다. 일치하는 얼굴에 대한 정보가 표시됩니다.

SearchFaces 작업 요청

얼굴 ID(얼굴 모음에 저장되는 각 얼굴은 얼굴 ID를 가짐)를 지정할 경우, SearchFaces는 지정된 얼굴 모음에서 유사한 얼굴을 검색합니다. 응답에는 검색하는 얼굴이 포함되지 않고 유사한 얼굴만 포함됩니다. SearchFaces는 기본적으로 알고리즘이 80% 이상의 유사성을 감지하는 얼굴을 반환합니다. 유사성은 얼굴이 입력 얼굴과 얼마나 일치하는지를 나타냅니다. 필요하다면 FaceMatchThreshold를 사용하여 다른 값을 지정할 수 있습니다.

```
{
    "CollectionId": "MyCollection",
    "FaceId": "0b683aed-a0f1-48b2-9b5e-139e9cc2a757",
    "MaxFaces": 2,
    "FaceMatchThreshold": 70
}
```

SearchFaces 작업 응답

작업은 발견된 얼굴 일치의 배열과 입력으로 제공한 얼굴 ID를 반환합니다.

```
{
    "SearchedFaceId": "7ecf8c19-5274-5917-9c91-1db9ae0449e2",
    "FaceMatches": [ list of face matches found ]
}
```

발견된 얼굴 일치마다 응답에는 유사성과 얼굴 메타데이터가 포함됩니다. 다음 예제 응답을 참조하십시오.

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

이미지를 사용하여 얼굴 검색

[SearchFacesByImage \(p. 311\)](#) 작업을 사용하여 제공된 이미지에서 가장 큰 얼굴과 일치하는 모음에서 얼굴을 검색할 수 있습니다.

자세한 내용은 [모음에 있는 얼굴 검색 \(p. 128\)](#) 단원을 참조하십시오.

이미지를 사용하여 모음에서 얼굴을 검색하려면(SDK)

1. 아직 설정하지 않았다면 다음과 같이 하십시오.

- a. AmazonRekognitionFullAccess 권한과 AmazonS3ReadOnlyAccess 권한을 가진 IAM 사용자를 만들어가 업데이트합니다. 자세한 내용은 [1단계: AWS 계정 설정 및 IAM 사용자 만들기 \(p. 11\)](#) 단원을 참조하십시오.
- b. AWS CLI와 AWS SDK를 설치하고 구성합니다. 자세한 내용은 [2단계: AWS CLI 및 AWS SDK 설정 \(p. 11\)](#) 단원을 참조하십시오.

2. 한 개 이상의 얼굴이 포함된 이미지를 S3 버킷에 업로드합니다.

이에 관한 지침은 Amazon Simple Storage Service 콘솔 사용 설명서의 [Amazon S3에 객체 업로드를 참조하십시오.](#)

3. 다음 예제를 사용하여 SearchFacesByImage 작업을 호출합니다.

Java

이 예제는 이미지에서 가장 큰 얼굴에 대한 정보를 표시합니다. 이 코드 예제는 FaceMatchThreshold 및 MaxFaces 파라미터를 지정하여 응답에서 반환되는 결과를 제한합니다.

다음 예제에서, collectionId 값을, 검색하려는 모음으로 바꾸고, bucket 값을 입력 이미지를 포함하는 버킷으로 바꾸고, photo 값을 입력 이미지로 바꿉니다.

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

AWS CLI

이 AWS CLI 명령은 `search-faces-by-image` CLI 작업에 대한 JSON 출력을 표시합니다. Bucket 값을 2단계에서 사용한 S3 버킷으로 바꿉니다. Name 값을 2단계에서 이미지 파일 이름으로 바꿉니다. collection-id 값을 검색할 모음으로 바꿉니다.

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

Python

이 예제는 이미지에서 가장 큰 얼굴에 대한 정보를 표시합니다. 이 코드 예제는 `FaceMatchThreshold` 및 `MaxFaces` 파라미터를 지정하여 응답에서 반환되는 결과를 제한합니다.

다음 예제에서, `collectionId` 값을 검색하려는 모음으로 바꾸고, `bucket` 값과 `photo` 값을 2단계에서 사용한 Amazon S3 버킷과 이미지 이름으로 바꿉니다.

```
#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

이 예제는 이미지에서 가장 큰 얼굴에 대한 정보를 표시합니다. 이 코드 예제는 FaceMatchThreshold 및 MaxFaces 파라미터를 지정하여 응답에서 반환되는 결과를 제한합니다.

다음 예제에서, collectionId 값을 검색하려는 모음으로 바꾸고, bucket 값과 photo 값을 2단계에서 사용한 Amazon S3 버킷과 이미지 이름으로 바꿉니다.

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

SearchFacesByImage 작업 요청

SearchFacesImageByImage에 대한 입력 파라미터는 검색할 모음과 소스 이미지 위치입니다. 이 예제에서 소스 이미지는 Amazon S3 버킷(S3Object)에 저장되어 있습니다. 또한 반환할 최대 얼굴 수(Maxfaces)와, 반환할 얼굴에 대해 일치해야 할 최소 신뢰도(FaceMatchThreshold)를 지정합니다.

```
{
    "CollectionId": "MyCollection",
    "Image": {
        "S3Object": {
            "Bucket": "bucket",
            "Name": "input.jpg"
        }
    },
    "MaxFaces": 2,
    "FaceMatchThreshold": 70
}
```

SearchFacesByImage 작업 응답

입력 이미지(.jpeg 또는 .png)가 주어진 경우, 작업은 먼저 입력 이미지에서 얼굴을 감지한 다음 지정된 얼굴 모음에서 유사한 얼굴을 감지합니다.

Note

입력 이미지에서 여러 얼굴이 감지되는 경우, 서비스는 감지된 가장 큰 얼굴을 얼굴 모음 감지에 사용합니다.

이 작업은 찾아낸 일치 얼굴의 배열과, 입력 얼굴에 대한 정보를 반환합니다. 경계 상자 등과 같은 정보와 신뢰도 값을 포함합니다. 신뢰도는 경계 상자의 얼굴 포함 기준이 되는 신뢰도 수준입니다.

SearchFacesByImage는 기본적으로 알고리즘이 80% 이상의 유사성을 감지하는 얼굴을 반환합니다. 유사성은 얼굴이 입력 얼굴과 얼마나 일치하는지를 나타냅니다. 필요하다면 FaceMatchThreshold를 사용하여 다른 값을 지정할 수 있습니다. 발견된 얼굴 일치마다 응답에는 유사성과 얼굴 메타데이터가 포함됩니다. 다음 예제 응답을 참조하십시오.

```
{
    "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  
}
```

저장된 비디오에서 얼굴 검색

저장된 비디오나 스트리밍 비디오에서 감지한 얼굴과 일치하는 얼굴을 얼굴 모음에서 찾아낼 수 있습니다. 이 단원에서는 저장된 비디오에서 얼굴을 검색하는 과정에 대해 다룹니다. 스트리밍 비디오에서 얼굴을 검색하는 방법에 대한 정보는 [스트리밍 비디오 작업 \(p. 84\)](#)를 참조하십시오.

먼저 [IndexFaces \(p. 287\)](#)를 사용하여 검색할 얼굴을 모음으로 인덱싱해야 합니다. 자세한 내용은 [모음에 얼굴 추가 \(p. 137\)](#) 단원을 참조하십시오.

Amazon Rekognition Video 얼굴 검색은 Amazon S3 버킷에 저장된 비디오를 분석하는 다른 Amazon Rekognition Video 작업과 동일한 비동기 워크플로우를 따릅니다. 저장된 비디오에서 얼굴 검색을 시작하려면 [StartFaceSearch \(p. 326\)](#)를 호출하고 검색하려는 모음의 ID를 입력합니다. Amazon Rekognition Video는 비디오 분석의 완료 상태를 Amazon Simple Notification Service 주제에 게시합니다. 비디오 분석이 성공적으로 완료되면 [GetFaceSearch \(p. 273\)](#)를 호출하여 검색 결과를 가져옵니다. 비디오 분석 시작 및 결과 가져오기에 대한 자세한 내용은 [Amazon Rekognition Video 작업 불러오기 \(p. 59\)](#) 단원을 참조하십시오.

다음 절차는 비디오에서 감지된 사람의 얼굴과 일치하는 얼굴 모음을 감지하는 방법을 보여 줍니다. 다음 절차에서는 비디오에서 일치되는 사람의 추적 데이터를 가져오는 방법을 보여 줍니다. 이 절차는 비디오 분석 요청의 완료 상태를 가져오기 위해 Amazon Simple Queue Service(Amazon SQS) 대기열을 사용하는 [Java](#) 또는 [Python](#)으로 [Amazon S3 버킷에 저장된 비디오 분석\(SDK\) \(p. 66\)](#)의 코드를 확장합니다.

비디오에서 일치하는 얼굴을 검색하려면(SDK)

1. [모음을 만듭니다 \(p. 128\)](#).
2. [얼굴을 모음으로 인덱싱합니다 \(p. 137\)](#).
3. [Java 또는 Python으로 Amazon S3 버킷에 저장된 비디오 분석\(SDK\) \(p. 66\)](#)을 수행합니다.
4. 3단계에서 만든 클래스 `VideoDetect`에 다음 코드를 추가합니다.

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)
throws Exception{

    StartFaceSearchRequest req = new StartFaceSearchRequest()
        .withCollectionId("CollectionId")
        .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 GetResultsFaceSearchCollection() 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);
}
```

4a. main 함수에서 다음 줄을

```
StartLabels(bucket,video);
```

다음으로 바꿉니다.

```
StartFaceSearchCollection(bucket,video);
```

4b. 다음 줄을

```
GetResultsLabels();
```

다음으로 바꿉니다.

```
GetResultsFaceSearchCollection();
```

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.)

def GetResultsFaceSearchCollection(self, jobId):
    maxResults = 10
    paginationToken = ''

    finished = False

    while finished == False:
        response = self.rek.get_face_search(JobId=jobId,
                                              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()
```

4a. main 함수에서 다음 줄을

```
response = self.rek.start_label_detection(Video={'S3Object': {'Bucket':
self.bucket, 'Name': self.video}}, NotificationChannel={'RoleArn':
self.roleArn, 'SNSTopicArn': self.topicArn})
```

다음으로 바꿉니다.

```
response = self.rek.start_face_search(Video={'S3Object':
{'Bucket':self.bucket,'Name':self.video}}, CollectionId='CollectionId',
NotificationChannel={'RoleArn':self.roleArn,
'SNSTopicArn':self.topicArn})
```

4b. 다음 줄을

```
self.GetResultsLabels(rekMessage['JobId'])
```

다음으로 바꿉니다.

```
self.GetResultsFaceSearchCollection(rekMessage['JobId'])
```

[Java 또는 Python으로 Amazon S3 버킷에 저장된 비디오 분석\(SDK\) \(p. 66\)](#) 이외에 비디오 예제를 이미 실행한 경우, 바꿀 함수 이름이 다릅니다.

5. CollectionId의 값을, 1단계에서 만든 모음의 이름으로 변경합니다.
6. 코드를 실행합니다. 입력 모음의 얼굴과 일치하는 비디오의 사람 목록이 표시됩니다. 일치하는 각 사람의 추적 데이터도 표시됩니다.

GetFaceSearch 작업 응답

다음은 GetFaceSearch의 JSON 응답 예제입니다.

응답에는 입력 모음의 얼굴과 얼굴이 일치하는 동영상에서 감지된 일련의 사람 Persons이 포함됩니다. 배열 요소 PersonMatch (p. 384)은 비디오에서 사람이 일치할 때마다 존재합니다. 각 PersonMatch에는 입력 모음에서 가져오는 일련의 얼굴 일치 FaceMatch (p. 363), 일치하는 사람에 대한 정보 PersonDetail (p. 382), 이 사람이 비디오에서 일치된 시간이 포함되어 있습니다.

```
{
    "JobStatus": "SUCCEEDED",
    "NextToken": "IJdbzkZfvBRqj8GPV82BPiZKkLOGCqDIsNZG/gQsEE5faTVK9JHOz/xxxxxxxxxxxxxx",
    "Persons": [
        {
            "FaceMatches": [
                {
                    "Face": {
```

```
"BoundingBox": {  
    "Height": 0.527472972869873,  
    "Left": 0.33530598878860474,  
    "Top": 0.2161169946193695,  
    "Width": 0.35503000020980835  
},  
    "Confidence": 99.90239715576172,  
    "ExternalImageId": "image.PNG",  
    "FaceId": "a2f2e224-bfaa-456c-b360-7c00241e5e2d",  
    "ImageId": "eb57ed44-8d8d-5ec5-90b8-6d190daff4c3"  
},  
    "Similarity": 98.40909576416016  
}  
],  
"Person": {  
    "BoundingBox": {  
        "Height": 0.8694444298744202,  
        "Left": 0.2473958283662796,  
        "Top": 0.10092592239379883,  
        "Width": 0.49427083134651184  
},  
    "Face": {  
        "BoundingBox": {  
            "Height": 0.23000000417232513,  
            "Left": 0.42500001192092896,  
            "Top": 0.16333332657814026,  
            "Width": 0.12937499582767487  
},  
        "Confidence": 99.97504425048828,  
        "Landmarks": [  
            {  
                "Type": "eyeLeft",  
                "X": 0.46415066719055176,  
                "Y": 0.2572723925113678  
},  
            {  
                "Type": "eyeRight",  
                "X": 0.5068183541297913,  
                "Y": 0.23705792427062988  
},  
            {  
                "Type": "nose",  
                "X": 0.49765899777412415,  
                "Y": 0.28383663296699524  
},  
            {  
                "Type": "mouthLeft",  
                "X": 0.487221896648407,  
                "Y": 0.3452930748462677  
},  
            {  
                "Type": "mouthRight",  
                "X": 0.5142884850502014,  
                "Y": 0.33167609572410583  
}  
],  
        "Pose": {  
            "Pitch": 15.966927528381348,  
            "Roll": -15.547388076782227,  
            "Yaw": 11.34195613861084  

```

```
        "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": 0.1583333134651184,  
"Width": 0.2369791716337204  
},  
"Face": {  
    "BoundingBox": {  
        "Height": 0.20000000298023224,  
        "Left": 0.029999999329447746,  
        "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": 2  
},  
"Timestamp": 0  
}.....  
],  
"VideoMetadata": {  
    "Codec": "h264",  
    "DurationMillis": 67301,  
    "Format": "QuickTime / MOV",  
    "FrameHeight": 1080,  
    "FrameRate": 29.970029830932617,  
    "FrameWidth": 1920  
}  
}
```

인물 추적

Amazon Rekognition Video는 비디오 속 인물을 추적하고 다음과 같은 정보를 제공할 수 있습니다.

- 추적 당시 비디오 프레임에 있는 인물의 위치.
- 왼쪽 눈의 위치와 같은 얼굴 표식(감지된 경우).

Amazon Rekognition Video의 저장된 비디오 속 인물 추적은 비동기식 작업입니다. 비디오 속 인물의 추적을 시작하려면 [StartPersonTracking \(p. 334\)](#)을 호출합니다. Amazon Rekognition Video는 비디오 분석의 완료 상태를 Amazon Simple Notification Service 주제에 게시합니다. 비디오 분석이 성공적으로 완료되면 [GetPersonTracking \(p. 282\)](#)을 호출하여 비디오 분석 결과를 가져오십시오. Amazon Rekognition Video API 작업 호출에 대한 자세한 내용은 [Amazon Rekognition Video 작업 불러오기 \(p. 59\)](#) 단원을 참조하십시오.

다음 절차에서는 Amazon S3 버킷에 저장된 비디오를 통해 인물을 추적하는 방법을 보여줍니다. 이 예제는 Amazon Simple Queue Service 대기열을 사용하여 비디오 분석 요청의 완료 상태를 가져오는 [Java 또는 Python으로 Amazon S3 버킷에 저장된 비디오 분석\(SDK\) \(p. 66\)](#)의 코드를 확장합니다.

Amazon S3 버킷에 저장된 비디오에서 인물을 감지하려면(SDK)

1. [Java 또는 Python으로 Amazon S3 버킷에 저장된 비디오 분석\(SDK\) \(p. 66\)](#)을 수행합니다.
2. 1단계에서 만든 클래스 VideoDetect에 다음 코드를 추가합니다.

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 StartPersons(String bucket, String video) throws Exception{

    int maxResults=10;
    String paginationToken=null;

    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 GetResultsPersons() throws Exception{
    int maxResults=10;
    String paginationToken=null;
    GetPersonTrackingResult personTrackingResult=null;

    do{
        if (personTrackingResult !=null){
            paginationToken = personTrackingResult.getNextToken();
        }
    }while(paginationToken !=null);
}
}
```

```

        }

        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);

}

```

2a. main 함수에서 다음 줄을

```
StartLabels(bucket,video);
```

다음으로 바꿉니다.

```
StartPersons(bucket,video);
```

2b. 다음 줄을

```
GetResultsLabels();
```

다음으로 바꿉니다.

```
GetResultsPersons();
```

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.)

def GetResultsPersons(self, jobId):
    maxResults = 10
    paginationToken = ''
    finished = False

    while finished == False:
        response = self.rek.get_person_tracking(JobId=jobId,
                                                MaxResults=maxResults,

```

```
        NextToken=paginationToken)

    print(response['VideoMetadata'][Codec'])
    print(str(response['VideoMetadata'][DurationMillis]))
    print(response['VideoMetadata'][Format])
    print(response['VideoMetadata'][FrameRate])

    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
```

2a. main 함수에서 다음 줄을

```
    response = self.rek.start_label_detection(Video={'S3Object': {'Bucket':
self.bucket, 'Name': self.video}}, NotificationChannel={'RoleArn':
self.roleArn, 'SNSTopicArn': self.topicArn})
```

다음으로 바꿉니다.

```
    response = self.rek.start_person_tracking(Video={'S3Object':
{'Bucket':self.bucket,'Name':self.video}}, NotificationChannel={'RoleArn':self.roleArn,
'SNSTopicArn':self.topicArn})
```

2b. 다음 줄을

```
    self.GetResultsLabels(rekMessage[JobId])
```

다음으로 바꿉니다.

```
    self.GetResultsPersons(rekMessage[JobId])
```

Note

[Java 또는 Python으로 Amazon S3 버킷에 저장된 비디오 분석\(SDK\)](#) (p. 66) 이외에 비디오 예제를 이미 실행한 경우, 바꿀 함수 이름이 다릅니다.

3. 코드를 실행합니다. 추적한 인물의 고유 식별자는 인물을 추적한 시간(초)과 함께 표시됩니다.

GetPersonTracking 작업 응답

GetPersonTracking은 추적한 인물에 대한 세부 정보와 비디오에서 해당 인물을 추적한 시간을 포함하는 [PersonDetection](#) (p. 383) 객체의 Persons 배열을 반환합니다.

SortBy 입력 파라미터를 사용하여 Persons를 정렬할 수 있습니다. 비디오에서 인물이 감지된 시간을 기준으로 요소를 정렬하려면 TIMESTAMP를 지정하십시오. 비디오에서 추적한 인물을 기준으로 정렬하려면 INDEX를 지정하십시오. 인물에 대한 각 결과 세트 내에서 요소는 추적의 정확성에 대한 정확도 기준 내림차순으로 정렬됩니다. 기본적으로 Persons은 TIMESTAMP를 기준으로 정렬되어 반환됩니다. 다음 예제는

GetPersonDetection의 JSON 응답입니다. 결과는 비디오 시작 이후 비디오에서 인물을 추적한 시간(밀리초)을 기준으로 정렬됩니다. 응답에서 다음에 유의하십시오.

- 인물 정보 – PersonDetection 배열 요소에는 감지된 인물에 대한 정보가 포함되어 있습니다. 예를 들어 인물이 감지된 시간(Timestamp), 감지 당시 비디오 프레임에 있는 인물의 위치(BoundingBox), 인물 감지의 정확성에 대한 Amazon Rekognition Video의 신뢰도(Confidence)입니다.

얼굴 특징은 인물을 추적한 모든 타임스탬프에서 반환되지 않습니다. 또한 상황에 따라 추적한 인물의 몸이 보이지 않을 수 있으며 이 경우 얼굴 위치만 반환됩니다.
- 페이지 정보 – 예제는 인물 감지 정보의 페이지 하나를 보여줍니다. GetPersonTracking의 MaxResults 입력 파라미터에서 반환할 인물 요소의 수를 지정할 수 있습니다. MaxResults 보다 많은 결과가 존재할 경우 GetPersonTracking은 결과의 다음 페이지를 가져올 때 사용되는 토큰(NextToken)을 반환합니다. 자세한 내용은 [Amazon Rekognition Video 분석 결과 가져오기 \(p. 61\)](#) 단원을 참조하십시오.
- 인덱스 – 비디오 전체에서 인물을 추적하기 위한 고유 식별자입니다.
- 비디오 정보 – 응답에는 GetPersonDetection에서 반환된 정보의 각 페이지에 있는 비디오 형식(VideoMetadata)에 관한 정보가 포함되어 있습니다.

```
{  
    "JobStatus": "SUCCEEDED",  
    "NextToken": "AcDymG0fSSoal6+BBYpka5wVlqttysSPP8VvWcujMDluj1QpFo/vf  
+mrMoqBGk8eUEiFlllR6g==",  
    "Persons": [  
        {  
            "Person": {  
                "BoundingBox": {  
                    "Height": 0.8787037134170532,  
                    "Left": 0.00572916679084301,  
                    "Top": 0.12129629403352737,  
                    "Width": 0.2166666865348816  
                },  
                "Face": {  
                    "BoundingBox": {  
                        "Height": 0.2000000298023224,  
                        "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
}
}
```

유명 인사 인식

Amazon Rekognition은 엔터테인먼트와 미디어, 스포츠, 재계, 정계 등 다양한 범주에 걸쳐 수천 명의 유명 인사 얼굴을 인식할 수 있습니다. Amazon Rekognition은 이미지와 저장된 비디오에서 유명 인사를 인식할 수 있습니다. 인식된 유명 인사에 대한 추가 정보를 얻을 수도 있습니다.

항목

- [이미지 속 유명 인사 인식 \(p. 173\)](#)
- [저장된 비디오 속 유명 인사 인식 \(p. 179\)](#)
- [유명 인사에 대한 정보 얻기 \(p. 183\)](#)

이미지 속 유명 인사 인식

이미지 속 유명 인사를 인식하고, 인식한 유명 인사에 대한 추가 정보를 얻으려면

[RecognizeCelebrities \(p. 304\)](#) 비 스토리지 API 작업을 사용하십시오. 예를 들어 정보 수집 타이밍이 중요한 소셜 미디어 또는 뉴스 및 엔터테인먼트 업계에서 `RecognizeCelebrities` 작업을 사용하여 한 이미지에서 무려 100명의 유명 인사를 식별하고 유명 인사 웹 페이지의 링크를 반환할 수 있습니다(해당하는 경우). Amazon Rekognition은 어떤 이미지에서 유명 인사가 감지되었는지 기억하지 못합니다. 애플리케이션에서 이 정보를 저장해야 합니다.

`RecognizeCelebrities`에서 반환한 유명 인사에 대한 추가 정보를 저장하지 않은 상태에서 정보 수집을 위해 이미지를 다시 분석하지 않으려면 [GetCelebrityInfo \(p. 257\)](#)를 사용합니다. `GetCelebrityInfo`를 호출하려면 Amazon Rekognition이 각 유명 인사에게 할당할 고유한 식별자가 필요합니다. 식별자는 이미지에서 인식된 각 유명 인사에 대한 `RecognizeCelebrities` 응답의 일부로 반환됩니다.

유명 인사 인식을 위해 처리할 이미지 모음이 많은 경우, [AWS Batch](#)를 사용해 백그라운드에서 일괄적으로 `RecognizeCelebrities`에 대한 호출을 처리하는 방법을 고려하십시오. 모음에 새 이미지를 추가하면, 이미지가 S3 버킷에 업로드될 때 AWS Lambda 함수를 사용해 `RecognizeCelebrities`를 호출하여 유명 인사를 인식할 수 있습니다.

RecognizeCelebrities 호출

입력 이미지는 AWS Command Line Interface(AWS CLI) 또는 AWS SDK를 사용하여 이미지 바이트 배열(base64 인코딩된 이미지 바이트) 또는 Amazon S3 객체로 제공할 수 있습니다. AWS CLI 절차에서는 .jpg 또는 .png 형식으로 이미지를 S3 버킷에 업로드합니다. AWS SDK for Java 절차에서는 로컬 파일 시스템에서 가져온 이미지를 사용합니다. 입력 이미지 사항에 대한 자세한 내용은 [이미지 작업 \(p. 35\)](#) 단원을 참조하십시오.

이 절차를 실행하려면 한 명 이상의 유명 인사 얼굴이 포함된 이미지 파일이 필요합니다.

이미지에서 유명 인사를 인식하려면

1. 아직 설정하지 않았다면 다음과 같이 하십시오.

- a. `AmazonRekognitionFullAccess` 권한과 `AmazonS3ReadOnlyAccess` 권한을 가진 IAM 사용자를 만들어가 업데이트합니다. 자세한 내용은 [1단계: AWS 계정 설정 및 IAM 사용자 만들기 \(p. 11\)](#) 단원을 참조하십시오.

- b. AWS CLI와 AWS SDK를 설치하고 구성합니다. 자세한 내용은 [2단계: AWS CLI 및 AWS SDK 설정 \(p. 11\)](#) 단원을 참조하십시오.
2. 다음 예제를 사용하여 `RecognizeCelebrities` 작업을 호출합니다.

Java

이 예제는 이미지에서 감지된 유명 인사에 대한 정보를 표시합니다.

`photo`의 값을, 하나 이상의 유명 인사 얼굴을 포함하는 이미지 파일의 경로와 파일 이름으로 변경합니다.

```
//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.");  
}
```

AWS CLI

이 AWS CLI 명령은 `recognize-celebrities` CLI 작업에 대한 JSON 출력을 표시합니다.

`bucketname`을 이미지가 저장된 Amazon S3 버킷의 이름으로 변경합니다. `input.jpg`를 하나 이상의 유명 인사 얼굴을 포함하는 이미지 파일의 이름으로 변경합니다.

```
aws rekognition recognize-celebrities \  
--image "S3Object={Bucket=bucketname,Name=input.jpg}"
```

Python

이 예제는 이미지에서 감지된 유명 인사에 대한 정보를 표시합니다.

`photo`의 값을, 하나 이상의 유명 인사 얼굴을 포함하는 이미지 파일의 경로와 파일 이름으로 변경합니다.

```
#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  
  
if __name__ == "__main__":  
    photo='moviestars.jpg'  
  
    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 '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
```

.NET

이 예제는 이미지에서 감지된 유명 인사에 대한 정보를 표시합니다.

photo의 값을, 하나 이상의 유명 인사 얼굴(.jpg 또는 .png 형식)을 포함하는 이미지 파일의 경로와 파일 이름으로 변경합니다.

```
//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.");
}
}
```

3. 표시되는 유명 인사 ID 중 하나의 값을 기록합니다. [유명 인사에 대한 정보 얻기 \(p. 183\)](#)에서 이 값이 필요할 것입니다.

RecognizeCelebrities 작업 요청

RecognizeCelebrities에 대한 입력은 이미지입니다. 이 예에서는 이미지가 이미지 바이트로 전달됩니다. 자세한 내용은 [이미지 작업 \(p. 35\)](#) 단원을 참조하십시오.

```
{
    "Image": {
        "Bytes": "/AoSiyvFpm....."
    }
}
```

RecognizeCelebrities 작업 응답

다음은 RecognizeCelebrities에 대한 예제 JSON 입력 및 출력입니다.

RecognizeCelebrities는 인식된 유명 인사의 배열과 인식되지 않는 얼굴의 배열을 반환합니다. 예제에서 다음 사항에 유의하십시오.

- 인식된 유명 인사 – Celebrities는 인식된 유명 인사의 배열입니다. 배열의 각 [Celebrity \(p. 346\)](#) 객체에는 유명 인사 이름과 관련 콘텐츠(예: 유명 인사의 IMDB 링크)를 가리키는 URL 목록이 포함됩니다. Amazon Rekognition은 애플리케이션에서 유명 인사의 얼굴이 이미지의 어느 위치에 있는지 파악하는 데 사용할 수 있는 [ComparedFace \(p. 350\)](#) 객체와 유명인의 고유한 식별자를 반환합니다. 나중에 [GetCelebrityInfo \(p. 257\)](#) API 작업으로 유명 인사 정보를 검색하려면 고유한 식별자를 사용합니다.
- 인식되지 않는 얼굴 – UnrecognizedFaces 알려진 유명 인사와 일치하지 않는 얼굴의 배열입니다. 배열의 각 [ComparedFace \(p. 350\)](#) 객체에는 이미지 속에서 얼굴을 찾는 데 사용할 수 있는 경계 상자(기타 정보와 함께)가 포함되어 있습니다.
- 이미지 방향 – OrientationCorrection은 이미지를 올바르게 표시하기 위해 사용할 수 있는 이미지 방향 정보입니다. 자세한 내용은 [이미지 방향 및 경계 상자 좌표 가져오기 \(p. 47\)](#) 단원을 참조하십시오.

```
{
    "CelebrityFaces": [
        "Face": {
            "BoundingBox": {
                "Height": 0.617123007774353,
                "Left": 0.15641026198863983,
                "Top": 0.10864841192960739,
                "Width": 0.3641025722026825
            },
            "Confidence": 99.99589538574219,
            "Landmarks": [
                {
                    "Type": "eyeLeft",
                    "X": 0.2837241291999817,
                    "Y": 0.3637104034423828
                }
            ]
        }
    ]
}
```

```
        },
        {
            "Type": "eyeRight",
            "X": 0.4091649055480957,
            "Y": 0.37378931045532227
        },
        {
            "Type": "nose",
            "X": 0.35267341136932373,
            "Y": 0.49657556414604187
        },
        {
            "Type": "mouthLeft",
            "X": 0.2786353826522827,
            "Y": 0.5455248355865479
        },
        {
            "Type": "mouthRight",
            "X": 0.39566439390182495,
            "Y": 0.5597742199897766
        }],
        "Pose": {
            "Pitch": -7.749263763427734,
            "Roll": 2.004552125930786,
            "Yaw": 9.012002944946289
        },
        "Quality": {
            "Brightness": 32.69192123413086,
            "Sharpness": 99.9305191040039
        }
    },
    "Id": "3Ir0du6",
    "MatchConfidence": 98.0,
    "Name": "Jeff Bezos",
    "Urls": ["www.imdb.com/name/nm1757263"]
}],
"OrientationCorrection": "ROTATE_0",
"UnrecognizedFaces": [
    {
        "BoundingBox": {
            "Height": 0.5345501899719238,
            "Left": 0.48461538553237915,
            "Top": 0.16949152946472168,
            "Width": 0.3153846263885498
        },
        "Confidence": 99.92860412597656,
        "Landmarks": [
            {
                "Type": "eyeLeft",
                "X": 0.5863404870033264,
                "Y": 0.36940744519233704
            },
            {
                "Type": "eyeRight",
                "X": 0.6999204754829407,
                "Y": 0.3769848346710205
            },
            {
                "Type": "nose",
                "X": 0.6349524259567261,
                "Y": 0.4804527163505554
            },
            {
                "Type": "mouthLeft",
                "X": 0.5872702598571777,
                "Y": 0.5535582304000854
            },
            {
                "Type": "mouthRight",
                "X": 0.6952020525932312,
                "Y": 0.5600858926773071
            }
        ],
        "Pose": {
            "Pitch": -7.386096477508545,
            "Roll": 2.304218292236328,
            "Yaw": -6.175624370574951
        }
    }
]
```

```
        },
        "Quality": {
            "Brightness": 37.16635513305664,
            "Sharpness": 99.9305191040039
        }
    }]
}
```

저장된 비디오 속 유명 인사 인식

Amazon Rekognition Video의 저장된 비디오 속 유명 인사 인식은 비동기 작업입니다. 저장된 비디오에서 유명 인사를 인식하려면 [StartCelebrityRecognition \(p. 315\)](#)을 사용하여 비디오 분석을 시작합니다. Amazon Rekognition Video는 비디오 분석의 완료 상태를 Amazon Simple Notification Service 주제에 게시합니다. 비디오 분석에 성공하면 [GetCelebrityRecognition \(p. 259\)](#)을 호출하여 분석 결과를 가져옵니다. 비디오 분석 시작 및 결과 가져오기에 대한 자세한 내용은 [Amazon Rekognition Video 작업 불러오기 \(p. 59\)](#) 단원을 참조 하십시오.

이 절차는 동영상 분석 요청의 완료 상태를 가져오기 위해 Amazon SQS 대기열을 사용하는 [Java 또는 Python으로 Amazon S3 버킷에 저장된 비디오 분석\(SDK\) \(p. 66\)](#)의 코드를 확장합니다. 이 절차를 실행하려면 한 명 이상의 유명 인사 얼굴이 포함된 비디오 파일이 필요합니다.

Amazon S3 버킷에 저장된 비디오에서 유명 인사를 감지하려면(SDK)

1. [Java 또는 Python으로 Amazon S3 버킷에 저장된 비디오 분석\(SDK\) \(p. 66\)](#)을 수행합니다.
2. 1단계에서 만든 클래스 VideoDetect에 다음 코드를 추가합니다.

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 StartCelebrities(String bucket, String video) throws Exception{

    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 GetResultsCelebrities() 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);

}
```

2a. main 함수에서 다음 줄을

```
StartLabels(bucket,video);
```

다음으로 바꿉니다.

```
StartCelebrities(bucket,video);
```

2b. 다음 줄을

```
GetResultsLabels();
```

다음으로 바꿉니다.

```
GetResultsCelebrities();
```

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.)
```

```
def GetResultsCelebrities(self, jobId):
    maxResults = 10
    paginationToken = ''
```

```
finished = False

while finished == False:
    response = self.rek.get_celebrity_recognition(JobId=jobId,
                                                   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
```

2a. main 함수에서 다음 줄을

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

                                         NotificationChannel={'RoleArn':
self.roleArn, 'SNSTopicArn': self.topicArn})
```

다음으로 바꿉니다.

```
response = self.rek.start_celebrity_recognition(Video={'S3Object':
{'Bucket':self.bucket,'Name':self.video}},

                                                NotificationChannel={'RoleArn':self.roleArn,
                                                'SNSTopicArn':self.topicArn})
```

2b. 다음 줄을

```
self.GetResultsLabels(rekMessage['JobId'])
```

다음으로 바꿉니다.

```
self.GetResultsCelebrities(rekMessage['JobId'])
```

Note

Java 또는 Python으로 Amazon S3 버킷에 저장된 비디오 분석(SDK) (p. 66) 이외에 비디오 예제를 이미 실행한 경우, 바꿀 함수 이름이 다릅니다.

3. 코드를 실행합니다. 비디오에서 인식된 유명 인사에 관한 정보가 표시됩니다.

GetCelebrityRecognition 작업 응답

다음은 JSON 응답의 예입니다. 응답에는 다음이 포함됩니다.

- 인식된 유명 인사 – `Celebrities` 비디오 속 유명 인사의 배열과 인식된 시간입니다. 비디오에서 유명 인사가 인식될 때마다 [CelebrityRecognition \(p. 349\)](#) 객체가 존재합니다. 각 `CelebrityRecognition`에는 인식된 유명 인사([CelebrityDetail \(p. 347\)](#))와 해당 유명 인사가 비디오에서 인식된 시간(Timestamp)에 대한 정보가 포함되어 있습니다. `Timestamp`은 비디오가 시작된 순간부터 밀리초 단위로 측정됩니다.
- `CelebrityDetail` – 인식된 유명 인사에 대한 정보가 들어 있습니다. 유명 인사 이름(Name), 식별자(ID) 및 관련 콘텐츠를 가리키는 URL 목록(Urls)이 포함됩니다. 또한 유명 인사의 신체에 대한 경계 상자, 인식의 정확성을 나타내는 Amazon Rekognition Video의 정확도 수준, 유명 인사의 얼굴에 대한 세부 정보, [FaceDetail \(p. 359\)](#)도 포함합니다. 나중에 관련 콘텐츠를 가져와야 하는 경우 [GetCelebrityInfo \(p. 257\)](#)와 함께 ID를 사용할 수 있습니다.
- `VideoMetadata` – 분석된 비디오에 대한 정보입니다.

```
{  
    "Celebrities": [  
        {  
            "Celebrity": {  
                "BoundingBox": {  
                    "Height": 0.8842592835426331,  
                    "Left": 0,  
                    "Top": 0.11574073880910873,  
                    "Width": 0.24427083134651184  
                },  
                "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",
    "Name": "Celeb A",
    "Urls": []
},
"Timestamp": 367
},.....
],
"JobStatus": "SUCCEEDED",
"NextToken": "XfxnZKiyMOGDhzBzYUhS5puM+g1IgezqFeYpv/H/+5noP/LmM57FitU AwSQ5D6G4AB/
PNwolrw==",
"VideoMetadata": {
    "Codec": "h264",
    "DurationMillis": 67301,
    "FileExtension": "mp4",
    "Format": "QuickTime / MOV",
    "FrameHeight": 1080,
    "FrameRate": 29.970029830932617,
    "FrameWidth": 1920
}
}
```

유명 인사에 대한 정보 얻기

이 절차에서는 [GetCelebrityInfo \(p. 257\)](#) API 작업을 사용하여 유명 인사 정보를 얻습니다. 유명 인사는 이 전 [the section called “RecognizeCelebrities” \(p. 304\)](#) 호출에서 반환된 유명 인사 ID를 사용하여 식별됩니다.

GetCelebrityInfo 호출

이 절차에는 Amazon Rekognition이 알고 있는 유명 인사의 유명 인사 ID도 필요합니다. [이미지 속 유명 인사 인식 \(p. 173\)](#)에서 기록해둔 유명 인사 ID를 사용합니다.

유명 인사 정보를 획득하려면(SDK)

1. 아직 설정하지 않았다면 다음과 같이 하십시오.
 - a. `AmazonRekognitionFullAccess` 권한과 `AmazonS3ReadOnlyAccess` 권한을 가진 IAM 사용자를 만들어가 업데이트합니다. 자세한 내용은 [1단계: AWS 계정 설정 및 IAM 사용자 만들기 \(p. 11\)](#) 단원을 참조하십시오.
 - b. AWS CLI와 AWS SDK를 설치하고 구성합니다. 자세한 내용은 [2단계: AWS CLI 및 AWS SDK 설정 \(p. 11\)](#) 단원을 참조하십시오.
2. 다음 예제를 사용하여 `GetCelebrityInfo` 작업을 호출합니다.

Java

이 예제는 유명 인사의 이름과 정보를 표시합니다.

`id`를 [이미지 속 유명 인사 인식 \(p. 173\)](#)에 표시된 유명 인사 ID 중 하나로 바꿉니다.

```
//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 = "nnnnnnnnn";

        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);
        }
    }
}
```

AWS CLI

이 AWS CLI 명령은 get-celebrity-info CLI 작업에 대한 JSON 출력을 표시합니다. ID를 [이미지 속 유명 인사 인식 \(p. 173\)](#)에 표시된 유명 인사 ID 중 하나로 바꿉니다.

```
aws rekognition get-celebrity-info --id ID
```

Python

이 예제는 유명 인사의 이름과 정보를 표시합니다.

*id*를 [이미지 속 유명 인사 인식 \(p. 173\)](#)에 표시된 유명 인사 ID 중 하나로 바꿉니다.

```
#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__":
    id="nnnnnnnnn"

    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)
```

.NET

이 예제는 유명 인사의 이름과 정보를 표시합니다.

[id를 이미지 속 유명 인사 인식 \(p. 173\)](#)에 표시된 유명 인사 ID 중 하나로 바꿉니다.

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

GetCelebrityInfo 작업 요청

다음은 GetCelebrityInfo에 대한 예제 JSON 입력 및 출력입니다.

GetCelebrityInfo에 대한 입력은 해당 유명 인사의 ID입니다.

```
{  
    "Id": "nnnnnnnn"  
}
```

GetCelebrityInfo는 요청한 유명 인사의 정보에 대한 링크 배열(Urls)을 반환합니다.

```
{  
    "Name": "Celebrity Name",
```

```
    "Urls": [  
        "www.imdb.com/name/nmnnnnnnnn"  
    ]  
}
```

비안전 콘텐츠 감지

Amazon Rekognition은 이미지나 비디오에 노골적이거나 선정적인 성인 콘텐츠가 포함되어 있는지 판단할 수 있습니다. Amazon Rekognition Image는 [DetectModerationLabels \(p. 251\)](#) 작업을 지원하여 이미지에 서 안전하지 않은 콘텐츠를 감지합니다. Amazon Rekognition Video는 [StartContentModeration \(p. 318\)](#) 및 [GetContentModeration \(p. 264\)](#)를 사용하여 안전하지 않은 콘텐츠를 비동기식으로 감지할 수 있습니다.

Amazon Rekognition은 계층적 분류를 사용하여 노골적이고 선정적인 콘텐츠의 범주를 레이블 지정합니다. 두 가지 최상위 범주는 노골적 누드와 선정적입니다. 각 최상위 범주에는 다수의 2단계 범주가 있습니다.

최상위 카테고리	2단계 범주
노골적인 나체	나체
	남성 나체 그래픽
	여성 나체 그래픽
	성행위
	부분적 누드
선정적	여성 수영복 또는 속옷
	남성 수영복 또는 속옷
	노출이 심한 의상

애플리케이션에 대한 이미지의 적합성을 결정합니다. 예를 들어 선정적인 성격의 이미지는 허용하되 누드가 포함된 이미지는 허용하지 않을 수 있습니다. 이미지를 필터링하려면 [DetectModerationLabels\(이미지\)](#)와 [GetContentModeration\(비디오\)](#)가 반환하는 [ModerationLabel \(p. 377\)](#) 레이블 배열을 사용합니다.

[ModerationLabel](#) 배열에는 이전 범주의 레이블 및 인식된 콘텐츠의 정확성에 대한 추정된 신뢰도가 포함되어 있습니다. 식별된 모든 2단계 레이블과 함께 최상위 레이블이 반환됩니다. 예를 들어 Rekognition은 신뢰도 점수가 높은 "노골적 누드"를 최상위 레이블로 반환할 수 있습니다. 이것으로 필터링에 충분할 수 있지만 필요하다면 "부분적 누드" 같은 2단계 레이블의 신뢰도 점수를 사용하여 더 세분화된 필터링을 할 수도 있습니다. 관련 예제는 [안전하지 않은 이미지 감지\(API\) \(p. 187\)](#) 단원을 참조하십시오.

Note

Rekognition Unsafe Image Detection API는 노골적이고 선정적인 성인 콘텐츠에 대한 인증 기관이 아니며 이에 대한 포괄적 필터가 되려는 의도도 없습니다. 나아가 Unsafe Image Detection API는 이미지에 불법적 콘텐츠(아동 포르노 등)나 비정상적인 성인 콘텐츠가 포함되어 있는지 여부를 감지하지 않습니다.

항목

- [안전하지 않은 이미지 감지\(API\) \(p. 187\)](#)
- [안전하지 않은 저장된 비디오 감지 \(p. 192\)](#)

안전하지 않은 이미지 감지(API)

[DetectModerationLabels \(p. 251\)](#) 작업을 사용하여 이미지에 노골적이거나 선정적인 성인 콘텐츠가 포함되어 있는지 파악합니다.

이미지에서 안전하지 않은 콘텐츠 감지(SDK)

이미지는 .jpg 또는 .png 형식이어야 합니다. 입력 이미지를 이미지 바이트 배열(base64 인코딩 이미지 바이트)로 제공하거나 Amazon S3 객체를 지정할 수 있습니다. 이 절차에서는 이미지(.jpg 또는 .png)를 S3 버킷에 업로드합니다.

이 절차를 실행하려면 AWS CLI 및 AWS SDK for Java 을 설치해야 합니다. 자세한 내용은 [Amazon Rekognition 시작하기 \(p. 10\)](#) 단원을 참조하십시오. 사용하는 AWS 계정에 Amazon Rekognition API에 대한 액세스 권한이 있어야 합니다. 자세한 내용은 [Amazon Rekognition API 권한: 작업, 권한 및 리소스 참조 \(p. 31\)](#) 단원을 참조하십시오.

이미지에서 조정 레이블(SDK)을 감지하려면

1. 아직 설정하지 않았다면 다음과 같이 하십시오.

- a. AmazonRekognitionFullAccess 권한과 AmazonS3ReadOnlyAccess 권한을 가진 IAM 사용자를 만들어가 업데이트합니다. 자세한 내용은 [1단계: AWS 계정 설정 및 IAM 사용자 만들기 \(p. 11\)](#) 단원을 참조하십시오.
- b. AWS CLI와 AWS SDK를 설치하고 구성합니다. 자세한 내용은 [2단계: AWS CLI 및 AWS SDK 설정 \(p. 11\)](#) 단원을 참조하십시오.

2. S3 버킷에 이미지를 업로드합니다.

이에 관한 지침은 Amazon Simple Storage Service 콘솔 사용 설명서의 [Amazon S3에 객체 업로드](#)를 참조하십시오.

3. 다음 예제를 사용하여 DetectModerationLabels 작업을 호출합니다.

Java

이 예제는 감지된 관리(moderation) 레이블 이름, 신뢰도 수준, 감지된 관리(moderation) 레이블의 상위 레이블을 출력합니다.

bucket 및 photo 값을 2단계에서 사용한 S3 버킷 이름과 이미지 파일 이름으로 바꿉니다.

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

AWS CLI

이 AWS CLI 명령은 detect-moderation-labels CLI 작업에 대한 JSON 출력을 표시합니다.

bucket 및 input.jpg을 2단계에서 사용한 S3 버킷 이름과 이미지 파일 이름으로 바꿉니다.

```
aws rekognition detect-moderation-labels \
--image '{"S3Object":{"Bucket":"'bucket'", "Name":"'input.jpg"}}"'
```

Python

이 예제는 감지된 관리(moderation) 레이블 이름, 신뢰도 수준, 감지된 관리(moderation) 레이블의 상위 레이블을 출력합니다.

bucket 및 photo 값을 2단계에서 사용한 S3 버킷 이름과 이미지 파일 이름으로 바꿉니다.

```
#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__":
    photo='moderate.png'
    bucket='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'])
```

.NET

이 예제는 감지된 관리(moderation) 레이블 이름, 신뢰도 수준, 감지된 관리(moderation) 레이블의 상위 레이블을 출력합니다.

bucket 및 photo 값을 2단계에서 사용한 S3 버킷 이름과 이미지 파일 이름으로 바꿉니다.

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

DetectModerationLabels 작업 요청

DetectModerationLabels에 대한 입력은 이미지입니다. 이 예제 JSON 입력에서는 Amazon S3 버킷에서 소스 이미지를 불러옵니다. MinConfidence는 Amazon Rekognition Image가 응답에 반환하기 위해 총 족해야 할 감지된 레이블의 최소 정확성 신뢰도 수준입니다.

```
{  
    "Image": {  
        "S3Object": {  
            "Bucket": "bucket",  
            "Name": "input.jpg"  
        }  
    },  
    "MinConfidence": 60  
}
```

DetectModerationLabels 작업 응답

DetectModerationLabels가 S3 버킷의 입력 이미지를 감지하거나, 이미지를 이미지 바이트로 직접 제공할 수 있습니다. 다음 예제는 DetectModerationLabels 호출로부터의 응답입니다.

다음 JSON 응답 예제에서 다음에 유의하십시오.

- 비안전 이미지 감지 정보 – 이 예제는 이미지에서 발견되는 노골적이거나 선정적인 콘텐츠의 중재 레이블 목록을 보여 줍니다. 이 목록에는 이미지에서 감지되는 최상위 레이블과 각각의 2수준 레이블이 포함됩니다.

레이블 – 각 레이블에는 이름, 해당 레이블이 정확한지에 대한 Amazon Rekognition의 신뢰도 추정, 상위 레이블의 이름이 있습니다. 최상위 레이블의 상위 이름은 ""입니다.

레이블 신뢰도 – 각 레이블에는 레이블이 올바른지에 대해 Amazon Rekognition이 가지고 있는 백분율 신뢰도를 나타내는 0에서 100 사이의 신뢰도 값이 있습니다. API 작업 요청에서 응답으로 반환될 레이블에 필요한 신뢰도 수준을 지정합니다.

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

안전하지 않은 저장된 비디오 감지

Amazon Rekognition Video의 저장된 비디오 속 안전하지 않은 콘텐츠 감지는 비동기 작업입니다. 안전하지 않은 콘텐츠 감지를 시작하려면 [StartContentModeration \(p. 318\)](#)을 호출합니다. Amazon Rekognition Video는 비디오 분석의 완료 상태를 Amazon Simple Notification Service 주제에 게시합니다. 비디오 분석이 성공적으로 완료되면 [GetContentModeration \(p. 264\)](#)을 호출하여 분석 결과를 가져옵니다. 비디오 분석 시작 및 결과 가져오기에 대한 자세한 내용은 [Amazon Rekognition Video 작업 블러오기 \(p. 59\)](#) 단원을 참조하십시오.

이 절차는 비디오 분석 요청의 완료 상태를 가져오기 위해 Amazon Simple Queue Service 대기열을 사용하는 [Java 또는 Python으로 Amazon S3 버킷에 저장된 비디오 분석\(SDK\) \(p. 66\)](#)의 코드를 확장합니다.

Amazon S3 버킷에 저장된 비디오에서 안전하지 않은 콘텐츠를 감지하려면(SDK)

1. [Java 또는 Python으로 Amazon S3 버킷에 저장된 비디오 분석\(SDK\) \(p. 66\)](#)을 수행합니다.
2. 1단계에서 만든 클래스 `VideoDetect`에 다음 코드를 추가합니다.

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 StartModerationLabels(String bucket, String video) throws  
Exception{  
  
    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 GetResultsModerationLabels() 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);

}
```

2a. main 함수에서 다음 줄을

```
StartLabels(bucket,video);
```

다음으로 바꿉니다.

```
StartModerationLabels(bucket,video);
```

2b. 다음 줄을

```
GetResultsLabels();
```

다음으로 바꿉니다.

```
GetResultsModerationLabels();
```

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.)

def GetResultsModerationLabels(self, jobId):
    maxResults = 10
    paginationToken = ''
    finished = False

    while finished == False:
        response = self.rek.get_content_moderation(JobId=jobId,
                                                    MaxResults=maxResults,
                                                    NextToken=paginationToken)

        print(response['VideoMetadata']['Codec'])
        print(str(response['VideoMetadata']['DurationMillis']))
        print(response['VideoMetadata']['Format'])
        print(response['VideoMetadata']['FrameRate'])

        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
```

2a. main 함수에서 다음 줄을

```
    response = self.rek.start_label_detection(Video={'S3Object': {'Bucket':
self.bucket, 'Name': self.video}},
                                         NotificationChannel={'RoleArn':
self.roleArn, 'SNSTopicArn': self.topicArn})
```

다음으로 바꿉니다.

```
    response = self.rek.start_content_moderation(Video={'S3Object':
{'Bucket':self.bucket,'Name':self.video}},
                                              NotificationChannel={'RoleArn':self.roleArn,
'SNSTopicArn':self.topicArn})
```

2b. 다음 줄을

```
    self.GetResultsLabels(rekMessage['JobId'])
```

다음으로 바꿉니다.

```
    self.GetResultsModerationLabels(rekMessage['JobId'])
```

Note

Java 또는 Python으로 Amazon S3 버킷에 저장된 비디오 분석(SDK) (p. 66) 이외에 비디오 예제를 이미 실행한 경우, 바꿀 함수 이름이 다릅니다.

3. 코드를 실행합니다. 비디오에서 감지된 안전하지 않은 콘텐츠 레이블 목록이 표시됩니다.

GetContentModeration 작업 응답

GetContentModeration 응답은 ContentModerationDetection (p. 353) 객체의 배열 ModerationLabels입니다. 배열에는 조절 레이블이 감지될 때마다 요소가 포함됩니다. ContentModerationDetectionObject 객체 내에 있는 ModerationLabel (p. 377)에는 선정적인 콘텐츠나 성인 콘텐츠가 감지된 항목에 대한 정보가 포함되어 있습니다. Timestamp는 레이블이 감지될 때 비디오의 시작 시간(밀리초)입니다. 레이블은 이미지 분석을 통해 감지된 조정 레이블과 동일한 방식으로 계층적 구조로 구성됩니다. 자세한 내용은 비안전 콘텐츠 감지 (p. 187) 단원을 참조하십시오.

다음은 GetContentModeration의 응답 예제입니다.

```
{
```

```
"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": "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
}
}
```

텍스트 감지

Amazon Rekognition Text in Image는 이미지의 텍스트를 감지하여 머신 판독 가능한 텍스트로 변환합니다. 다음 예제처럼 머신 판독 가능한 텍스트를 솔루션을 구현할 수 있습니다.

- 시각적 검색. 예를 들어 동일한 텍스트를 포함하는 이미지를 검색하고 표시합니다.
- 콘텐츠 분석 정보. 예를 들어, 추출된 비디오 프레임에서 인식되는 텍스트에서 발생하는 주제에 대한 분석 정보를 제공합니다. 애플리케이션은 뉴스, 스포츠 경기 점수, 운동 선수 번호 및 캡션과 같은 관련 콘텐츠에서 인식된 텍스트를 검색할 수 있습니다.
- 탐색. 예를 들어 레스토랑, 상점 또는 거리 표지의 이름을 인식하는 시각 장애인을 위한 음성 지원 모바일 앱을 개발합니다.
- 공공 안전과 운송 지원. 예를 들어 교통 카메라 이미지에서 자동차 번호판 번호를 감지합니다.
- 필터링. 예를 들어, 이미지에서 개인 식별 정보를 필터링합니다.

[DetectText \(p. 254\)](#)는 .jpeg 또는 .png 형식의 이미지를 감지하고 고도로 정형화된 이미지를 포함한 대부분의 글꼴을 지원합니다. 텍스트를 감지한 후 DetectText는 감지된 단어와 텍스트 줄을 표현하고 이들 사이의 관계를 보여줍니다. DetectText API는 텍스트가 이미지의 어디에 있는지도 알려줍니다.

다음 이미지를 고려하십시오.



파란색 상자는 DetectText 작업이 반환하는 텍스트의 위치와 감지된 텍스트에 대한 정보를 나타냅니다. 텍스트가 감지되려면 텍스트가 가로 축의 +/- 30도 방향 내에 있어야 합니다. DetectText는 인식된 텍스트를 단어 또는 텍스트 줄로 분류합니다.

단어는 공백으로 구분되지 않는 하나 이상의 ISO 기본 라틴 문자입니다. DetectText는 이미지에서 최대 50개 단어를 감지할 수 있습니다.

줄은 동등하게 간격을 둔 단어로 구성된 문자열입니다. 줄은 반드시 완성된 문장이 아니어도 됩니다. 예를 들어, 운전 면허증 번호가 줄로 감지됩니다. 줄은 그 뒤에 정렬된 텍스트가 없을 때 끝납니다. 또한 단어의 길이에 비례하여 단어 사이에 큰 간격이 있을 때 줄이 끝납니다. 즉, 단어 사이의 간격에 따라 Amazon Rekognition은 같은 방향으로 정렬된 텍스트에서 여러 줄을 감지할 수 있습니다. 마침표는 줄의 끝을 나타내지 않습니다. 문장이 여러 줄에 걸쳐 있는 경우 DetectText 작업은 여러 줄을 반환합니다.

Amazon Rekognition은 숫자뿐 아니라 @, /, \$, %, -, _, +, *, #와 같은 일반 기호도 감지할 수 있습니다.

문제 해결 예는 [이미지에서 텍스트 감지 \(p. 196\)](#) 단원을 참조하십시오.

이미지에서 텍스트 감지

입력 이미지를 이미지 바이트 배열(base64 인코딩 이미지 바이트) 또는 Amazon S3 객체로 제공할 수 있습니다. 이 절차에서는 S3 버킷에 .jpeg 또는 .png 이미지를 업로드하고 파일 이름을 지정합니다.

이미지(API)에서 텍스트를 감지하려면

- 아직 설정하지 않았다면 다음과 같이 하십시오.

- a. AmazonRekognitionFullAccess 권한과 AmazonS3ReadOnlyAccess 권한을 가진 IAM 사용자를 만들어가 업데이트합니다. 자세한 내용은 [1단계: AWS 계정 설정 및 IAM 사용자 만들기 \(p. 11\)](#) 단원을 참조하십시오.
- b. AWS CLI와 AWS SDK를 설치하고 구성합니다. 자세한 내용은 [2단계: AWS CLI 및 AWS SDK 설정 \(p. 11\)](#) 단원을 참조하십시오.

2. 텍스트를 포함하는 이미지를 S3 버킷에 업로드합니다.

이에 관한 지침은 Amazon Simple Storage Service 콘솔 사용 설명서의 [Amazon S3에 객체 업로드](#)를 참조하십시오.

3. 다음 예제를 사용하여 DetectText 작업을 호출합니다.

Java

다음 예제 코드는 감지된 결과, 이미지에서 감지되었던 단어를 표시합니다.

bucket 및 photo의 값을 2단계에서 사용한 Amazon S3 버킷과 이미지의 이름으로 바꿉니다.

```
//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());  
            }  
        } catch (AmazonRekognitionException e) {  
            System.out.println("Error: " + e.getMessage());  
        }  
    }  
}
```

```
        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();
}
}
```

AWS CLI

이 AWS CLI 명령은 detect-text CLI 작업에 대한 JSON 출력을 표시합니다.

Bucket 및 Name의 값을 2단계에서 사용한 Amazon S3 버킷과 이미지의 이름으로 바꿉니다.

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

Python

다음 예제 코드는 감지된 줄과, 이미지에서 감지된 단어를 표시합니다.

bucket 및 photo의 값을 2단계에서 사용한 Amazon S3 버킷과 이미지의 이름으로 바꿉니다.

```
#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'
    photo='inputtext.jpg'

    client=boto3.client('rekognition')

    response=client.detect_text(Image={'S3Object':{'Bucket':bucket,'Name':photo}})

    textDetections=response['TextDetections']
    print response
    print 'Matching faces'
    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
```

.NET

다음 예제 코드는 감지된 줄과, 이미지에서 감지된 단어를 표시합니다.

bucket 및 photo의 값을 2단계에서 사용한 Amazon S3 버킷과 이미지의 이름으로 바꿉니다.

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

DetectText 작업 요청

DetectText 작업에서는 입력 이미지를 byte64로 인코딩된 바이트 배열 또는 Amazon S3 버킷에 저장된 이미지로 제공합니다. 다음 예제 JSON 요청은 Amazon S3 버킷에서 불러온 이미지를 표시합니다.

```
{  
    "Image": {  
        "S3Object": {  
            "Bucket": "bucket",
```

```
        "Name": "inputtext.jpg"  
    }  
}
```

DetectText 작업 응답

DetectText 작업은 이미지를 분석하고 TextDetections 배열을 반환합니다. 이때 각 요소 ([TextDetection \(p. 394\)](#))는 이미지에서 감지된 줄이나 단어를 나타냅니다. 각 요소마다 DetectText는 다음 정보를 반환합니다.

- 감지된 텍스트(DetectedText)
- 단어와 줄의 관계(Id 및 ParentId)
- 이미지에서 텍스트의 위치(Geometry)
- 정확도 Amazon Rekognition은 감지된 텍스트와 경계 상자의 정확성에 있습니다(Confidence)
- 감지된 텍스트(Type)의 유형

감지된 텍스트

각 TextDetection 요소는 DetectedText 필드에서 인식된 텍스트(단어 또는 줄)를 포함합니다. 반환된 텍스트에는 단어를 인식할 수 없도록 만드는 문자가 포함될 수 있습니다. 예를 들어, Cat가 아닌 C@t가 있기도 합니다. TextDetection 요소가 텍스트 또는 단어로 구성된 줄을 나타내는지 확인하려면 Type 필드를 사용합니다.

각 TextDetection 요소에는 감지된 텍스트와 텍스트를 둘러싼 경계 상자의 정확도에 대한 Amazon Rekognition의 정확도를 나타내는 백분율 값이 포함됩니다.

단어와 줄의 관계

각 TextDetection 요소에는 식별자 필드 Id가 있습니다. Id는 줄에서 단어의 위치를 나타냅니다. 요소가 단어인 경우, 상위 식별자 ParentId는 단어가 감지되는 줄을 확인해 줍니다. 줄의 ParentId는 null입니다. 예를 들어 선행 이미지의 "but keep"이라는 줄에는 Id 및 ParentId 값이 있습니다.

Text	ID	상위 ID
but keep	3	
but	8	3
Keep	9	3

이미지에서 텍스트의 위치

이미지에서 인식된 텍스트의 위치를 확인하려면 DetectText가 반환하는 경계 상자([Geometry \(p. 367\)](#)) 정보를 사용합니다. Geometry 객체에는 감지된 선과 단어에 대한 두 가지 유형의 경계 상자 정보가 있습니다.

- [BoundingBox \(p. 344\)](#) 객체의 축으로 정렬된 거친 직사각형 윤곽
- [Point \(p. 385\)](#) 배열에서 여러 개의 X와 Y 좌표로 구성된, 세분화된 다각형

테두리 상자와 다각형 좌표는 원본 이미지의 텍스트 위치를 나타냅니다. 좌표 값은 전체 이미지 크기의 비율입니다. 자세한 내용은 [BoundingBox \(p. 344\)](#) 단원을 참조하십시오.

DetectText 작업의 다음 JSON 응답은 다음 이미지에서 감지된 단어와 줄을 표시합니다.



```
{  
    "TextDetections": [  
        {  
            "Confidence": 90.54900360107422,  
            "DetectedText": "IT'S",  
            "Geometry": {  
                "BoundingBox": {  
                    "Height": 0.10317354649305344,  
                    "Left": 0.6677391529083252,  
                    "Top": 0.17569075524806976,  
                    "Width": 0.15113449096679688  
                },  
                "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": 0.08413040637969971
    },
    "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"
}
]
```

모니터링

Amazon Rekognition을 모니터링하려면 Amazon CloudWatch와 AWS CloudTrail을 사용합니다. 이 단원에서는 Rekognition 모니터링을 설정하는 방법에 대한 정보와 Rekognition 측정치 참조 콘텐츠를 제공합니다.

항목

- [Rekognition 모니터링 \(p. 207\)](#)
- [Rekognition의 CloudWatch 측정치 \(p. 210\)](#)
- [AWS CloudTrail을 사용하여 Amazon Rekognition API 호출 로깅 \(p. 211\)](#)

Rekognition 모니터링

CloudWatch를 사용하면 개별 Rekognition 작업에 대한 측정치 또는 계정에 대한 전역 Rekognition 측정치를 가져올 수 있습니다. 측정치를 사용하여 Rekognition 기반 솔루션의 상태를 추적하고 하나 이상의 측정치가 정의된 임계값을 벗어날 때 통보하도록 경보를 설정할 수 있습니다. 예를 들어 발생한 서버 오류 수에 대한 측정치나 감지된 얼굴 수에 대한 측정치를 볼 수 있습니다. 또한 특정 Rekognition 작업이 성공한 횟수에 대한 측정치도 볼 수 있습니다. 측정치를 보려면 [Amazon CloudWatch](#), [Amazon AWS Command Line Interface](#) 또는 [CloudWatch API](#)를 사용하면 됩니다.

또한 Rekognition 콘솔을 사용하여 선택한 기간 동안 집계된 측정치를 볼 수 있습니다. 자세한 내용은 [연습 4: 집계 측정치 보기\(콘솔\) \(p. 23\)](#) 단원을 참조하십시오.

Rekognition에 CloudWatch 측정치 사용

측정치를 사용하려면 다음 정보를 지정해야 합니다.

- 측정치 차원 또는 차원 없음. 차원은 지표를 고유하게 식별하는 데 도움이 되는 이름-값 페어입니다. Rekognition에는 Operation이라는 하나의 차원이 있습니다. 이는 특정 작업에 대한 측정치를 제공합니다. 차원을 지정하지 않으면 측정치의 범위가 계정 내의 모든 Rekognition 작업으로 지정됩니다.
- `UserErrorCount`와 같은 지표 이름.

AWS Management 콘솔, AWS CLI 또는 CloudWatch API를 사용하여 Rekognition의 모니터링 데이터를 가져올 수 있습니다. Amazon AWS 소프트웨어 개발 키트(SDK) 또는 CloudWatch API 도구 중 하나를 통해 CloudWatch API를 사용할 수도 있습니다. 콘솔에는 CloudWatch API의 원시 데이터를 근거로 일련의 그래프가 표시됩니다. 필요에 따라 콘솔에 표시되거나 API에서 가져온 그래프를 사용하는 것이 더 나을 수 있습니다.

다음은 몇 가지 일반적인 지표 사용 사례입니다. 모든 사용 사례를 망라한 것은 아니지만 시작하는 데 참고가 될 것입니다.

방법	관련 지표
인식되는 얼굴 수를 추적하려면 어떻게 해야 합니까?	<code>DetectedFaceCount</code> 측정치의 <code>Sum</code> 통계를 모니터링합니다.

방법	관련 지표
내 애플리케이션이 초당 최대 요청 수에 도달했는지 여부를 어떻게 알 수 있습니까?	ThrottledCount 측정치의 Sum 통계를 모니터링 합니다.
요청 오류는 어떻게 모니터링할 수 있습니까?	UserErrorCount 측정치의 Sum 통계를 사용합니다.
총 요청 수를 찾으려면 어떻게 해야 합니까?	ResponseTime 측정치의 ResponseTime 및 Data Samples 통계를 사용합니다. 여기에는 오류를 초래하는 모든 요청이 포함됩니다. 성공한 작업 호출만 보려면 SuccessfulRequestCount 측정치를 사용합니다.
Rekognition 작업 호출의 지연 시간은 어떻게 모니터링할 수 있습니까?	ResponseTime 측정치를 사용합니다.
IndexFaces가 Rekognition 모음에 성공적으로 얼굴을 추가한 횟수를 모니터링하려면 어떻게 해야 합니까?	SuccessfulRequestCount 측정치 및 IndexFaces 작업을 사용하여 Sum 통계를 모니터링합니다. Operation 차원을 사용하여 작업과 측정치를 선택합니다.

CloudWatch를 사용하여 Rekognition을 모니터링하려면 적절한 CloudWatch 권한이 있어야 합니다. 자세한 내용은 [Amazon CloudWatch 인증 및 액세스 제어](#) 단원을 참조하십시오.

Rekognition 측정치에 액세스

다음 예제는 CloudWatch 콘솔, AWS CLI 및 CloudWatch API를 사용하여 Rekognition 측정치에 액세스하는 방법을 보여 줍니다.

지표를 보려면(콘솔)

1. <https://console.aws.amazon.com/cloudwatch/>에서 CloudWatch 콘솔을 엽니다.
2. [Metrics]를 선택하고 [All Metrics] 탭을 선택한 후 [Rekognition]을 선택합니다.
3. [Metrics with no dimensions]를 선택한 후 측정치를 선택합니다.

예를 들어 얼마나 많은 얼굴이 감지되었는지 측정하려면 [DetectedFace] 측정치를 선택합니다.

4. 날짜 범위 값을 선택합니다. 측정치 개수는 그래프에 표시됩니다.

일정 기간 동안 성공적으로 이루어진 **DetectFaces** 작업 호출 측정치를 보려면(CLI)

- AWS CLI를 열고 다음 명령을 입력합니다.

```
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
```

이 예제는 일정 기간 동안 성공적으로 이루어진 DetectFaces 작업 호출을 보여 줍니다. 자세한 내용은 [get-metrics-statistics](#)를 참조하십시오.

측정치에 액세스하려면(CloudWatch API)

- [GetMetricStatistics](#)을 호출합니다. 자세한 내용은 [Amazon CloudWatch API Reference](#)를 참조하십시오.

경보 만들기

경보가 상태를 변경할 때 Amazon Simple Notification Service(Amazon SNS) 메시지를 보내는 CloudWatch 경보를 만들 수 있습니다. 경보는 지정한 기간에 단일 메트릭을 감시하고 여러 기간에 지정된 임계값에 대한 메트릭 값을 기준으로 작업을 하나 이상 수행합니다. 이 작업은 Amazon SNS 주제나 Auto Scaling 정책으로 전송되는 알림입니다.

경보는 지속적인 상태 변경에 대해서만 작업을 호출합니다. CloudWatch 경보는 단순히 특정 상태에 있다고 해서 작업을 호출하지 않습니다. 상태가 변경되어 지정된 기간 수 동안 유지되어야 합니다.

경보를 설정하려면(콘솔)

1. AWS Management 콘솔에 로그인한 다음 <https://console.aws.amazon.com/cloudwatch/>에서 CloudWatch 콘솔을 엽니다.
2. Create Alarm을 선택합니다. 그러면 [Create Alarm Wizard]가 시작됩니다.
3. [Metrics with no dimensions] 측정치 목록에서 [Rekognition Metrics]를 선택한 후 측정치를 선택합니다.
예를 들어 감지된 얼굴의 최대 수에 대한 경보를 설정하려면 [DetectedFaceCount]를 선택합니다.
4. [Time Range] 영역에서 호출한 얼굴 감지 작업이 포함된 날짜 범위 값을 선택합니다. [Next(다음)]를 선택합니다.
5. [Name]과 [Description]을 입력합니다. [Whenever]에서 [>=]를 선택하고 원하는 최대값을 입력합니다.
6. 경보 상태에 도달하면 CloudWatch에서 이메일을 보내도록 하려면 [Whenever this alarm:]에서 [State is ALARM]을 선택합니다. 기존 Amazon SNS 주제에 경보를 전송하려면 [Send notification to:]에서 기존 SNS 주제를 선택합니다. 새 이메일 구독 목록에 이름과 이메일 주소를 설정하려면 [Create topic]을 선택합니다. 나중에 경보를 설정하는 데 사용할 수 있도록 CloudWatch가 목록을 저장하고 필드에 표시합니다.

Note

[Create topic]을 사용하여 새 Amazon SNS 주제를 만드는 경우, 의도한 수신자가 알림을 받기 전에 이메일 주소를 확인해야 합니다. Amazon SNS는 경보가 경보 상태에 진입할 때만 이메일을 전송합니다. 이러한 경보 상태 변경이 이메일 주소 확인 전에 발생할 경우, 의도된 수신자는 알림을 받지 못합니다.

7. [Alarm Preview] 섹션에서 경보를 미리 봅니다. [Create Alarm]을 선택합니다.

경보를 설정하려면(AWS CLI)

- AWS CLI를 열고 다음 명령을 입력합니다. `alarm-actions` 파라미터의 값을 변경하면 이전에 만든 Amazon SNS 주제를 참조할 수 있습니다.

```
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
```

이 예제는 5분 이내에 10회 이상 사용자 오류가 발생하는 경우의 경보를 생성하는 방법을 보여 줍니다. 자세한 내용은 [put-metrics-alarm](#) 단원을 참조하십시오.

경보를 설정하려면(CloudWatch API)

- [PutMetricAlarm](#)을 호출합니다. 자세한 내용은 [Amazon CloudWatch API Reference](#)를 참조하십시오.

Rekognition의 CloudWatch 측정치

이 단원에는 Amazon Rekognition에 사용할 수 있는 Amazon CloudWatch 측정치 및 Operation 차원에 대한 정보가 나와 있습니다.

Rekognition 콘솔에서 Rekognition 측정치의 집계 확인도 볼 수 있습니다. 자세한 내용은 [연습 4: 집계 측정치 보기\(콘솔\) \(p. 23\)](#) 단원을 참조하십시오.

Rekognition의 CloudWatch 측정치

다음 표에는 Rekognition 측정치가 요약되어 있습니다.

지표	설명
SuccessfulRequestCount	<p>성공한 요청 수. 성공적 요청의 응답 코드 범위는 200 - 299입니다.</p> <p>단위: 수</p> <p>유효한 통계: Sum, Average</p>
ThrottledCount	<p>조정된 요청 수. Rekognition은 계정에 대해 설정된 초당 트랜잭션 한도 이상의 요청이 수신되면 요청을 제한합니다. 계정에 대해 설정된 한도가 자주 초과되면 한도 증가를 요청할 수 있습니다. 증가를 요청하려면 AWS 서비스 한도를 참조하십시오.</p> <p>단위: 수</p> <p>유효한 통계: Sum, Average</p>
ResponseTime	<p>Rekognition이 응답을 계산하는 시간(밀리초).</p> <p>단위:</p> <ol style="list-style-type: none">1. Data Samples 통계 개수2. Average 통계의 밀리초 <p>유효한 통계: Data Samples, Average</p> <p>Note</p> <p>ResponseTime 측정치는 Rekognition 측정치 창에 포함되지 않습니다.</p>
DetectedFaceCount	<p>IndexFaces 또는 DetectFaces 작업으로 감지된 얼굴의 수.</p> <p>단위: 수</p> <p>유효한 통계: Sum, Average</p>
DetectedLabelCount	<p>DetectLabels 작업을 사용하여 감지된 레이블 수.</p> <p>단위: 수</p> <p>유효한 통계: Sum, Average</p>
ServerErrorCount	<p>서버 오류 수. 서버 오류의 응답 코드 범위는 500 - 599입니다.</p> <p>단위: 수</p>

지표	설명
	유효한 통계: Sum, Average
UserErrorCount	사용자 오류 수(잘못된 파라미터, 잘못된 이미지, 권한 없음 등). 사용자 오류의 응답 코드 범위는 400~499입니다. 단위: 수 유효한 통계: Sum, Average

Rekognition의 CloudWatch 차원

작업별 측정치를 검색하려면 Rekognition 네임스페이스를 사용하고 operation 차원을 제공합니다. 차원에 대한 자세한 내용은 Amazon CloudWatch 사용 설명서의 [차원](#) 단원을 참조하십시오.

AWS CloudTrail을 사용하여 Amazon Rekognition API 호출 로깅

Amazon Rekognition는 Amazon Rekognition에서 사용자, 역할 또는 AWS 서비스가 수행한 작업에 대한 레코드를 제공하는 서비스인 AWS CloudTrail과 통합됩니다. 추적을 생성하면 Amazon S3 버킷, Amazon CloudWatch Logs 및 Amazon CloudWatch Events로 CloudTrail 이벤트를 지속적으로 배포하도록 할 수 있습니다. Amazon Rekognition은 CloudTrail의 이벤트 기록 기능에도 통합되어 있습니다. Amazon Rekognition용 API가 이벤트 기록에서 지원되는 경우에는 CloudTrail에 어떤 로그도 구성하지 않은 경우라도 이벤트 기록에서 지난 90일간의 CloudTrail 콘솔 Amazon Rekognition 이벤트를 볼 수 있습니다. CloudTrail에서 수집하는 정보를 사용하여 Amazon Rekognition에 어떤 요청이 이루어졌는지, 어떤 IP 주소에서 요청했는지, 누가 언제 요청했는지 및 추가 세부 정보를 확인할 수 있습니다.

그 구성 및 활성화 방법을 포함하여 CloudTrail에 대한 자세한 내용은 [AWS CloudTrail User Guide](#) 단원을 참조하십시오.

CloudTrail의 Amazon Rekognition 정보

Amazon Rekognition 서비스는 CloudTrail 로그 파일의 이벤트로 다음 작업의 로깅을 지원합니다.

- the section called “CreateCollection” (p. 224)
- the section called “DeleteCollection” (p. 230)
- the section called “CreateStreamProcessor” (p. 227)
- the section called “DeleteStreamProcessor” (p. 234)
- the section called “DescribeStreamProcessor” (p. 239)
- the section called “ListStreamProcessors” (p. 301)
- the section called “ListCollections” (p. 295)

모든 이벤트 및 로그 항목에는 요청을 생성한 사용자에 대한 정보가 들어 있습니다. 자격 증명 정보를 이용하면 다음을 쉽게 판단할 수 있습니다.

- 요청을 루트로 했는지, IAM 사용자 자격 증명으로 했는지.
- 역할 또는 연합된 사용자에 대한 임시 보안 자격 증명을 사용하여 요청이 생성되었는지 여부.
- 요청을 다른 AWS 서비스로 했는지.

자세한 정보는 [CloudTrail userIdentity 요소](#)를 참조하십시오.

CloudTrail은 계정 생성 시 AWS 계정에서 활성화됩니다. 활동이 Amazon Rekognition에서 이루어지면 해당 활동이 이벤트 기록의 다른 AWS 서비스 이벤트와 함께 CloudTrail 이벤트 로그에 기록됩니다. 이를 통해 AWS 계정에서 이루어진 지난 90일간의 지원 활동을 확인, 검색 및 다운로드할 수 있습니다. 자세한 내용은 [CloudTrail 이벤트 기록에서 이벤트 보기](#) 및 [CloudTrail 이벤트 기록에서 지원하는 서비스](#)를 참조하십시오.

추적을 생성하고 원하는 기간만큼 Amazon S3 버킷에 로그 파일을 저장할 수 있을 뿐 아니라, Amazon S3 수명 주기 규칙을 정의하여 자동으로 로그 파일을 보관하거나 삭제할 수 있습니다. 기본적으로 로그 파일은 Amazon S3 서버 측 암호화(SSE)를 통해 암호화됩니다.

새 로그 파일이 전달되면 Amazon SNS 알림을 게시하도록 CloudTrail을 구성하여 로그 파일 전송 시 알림을 받을 수 있습니다. 자세한 내용은 [CloudTrail용 Amazon SNS 알림 설정](#)을 참조하십시오.

또한 여러 AWS 리전 및 여러 AWS 계정의 Amazon Rekognition 로그 파일을 하나의 Amazon S3 버킷으로 통합할 수도 있습니다.

자세한 내용은 [여러 리전에서 CloudTrail 로그 파일 받기](#)와 [여러 계정에서 CloudTrail 로그 파일 받기](#)를 참조하십시오.

예: Amazon Rekognition 로그 파일 항목

추적은 지정한 Amazon S3 버킷에 이벤트를 로그 파일로 제공할 수 있도록 해주는 구성입니다. CloudTrail은 그 파일에는 하나 이상의 로그 항목이 포함됩니다. 이벤트는 어떤 소스로부터의 단일 요청을 나타내며 요청된 작업, 작업 날짜와 시간, 요청 파라미터 등에 대한 정보가 들어 있습니다. CloudTrail 로그 파일은 퍼블릭 API 호출의 주문 스택 트레이스가 아니기 때문에 특정 순서로 표시되지 않습니다.

다음은 CloudTrail 로그 항목과 CreateCollection, DeleteCollection, CreateStreamProcessor, DeleteStreamProcessor, DescribeStreamProcessor, ListStreamProcessors, ListCollections 등의 API에 대한 작업을 보여 주는 예제입니다.

```
{  
    "Records": [  
        {  
            "eventVersion": "1.05",  
            "userIdentity": {  
                "type": "IAMUser",  
                "principalId": "EX_PRINCIPAL_ID",  
                "arn": "arn:aws:iam::111122223333:user/Alice",  
                "accountId": "111122223333",  
                "accessKeyId": "EXAMPLE_KEY_ID",  
                "userName": "Alice"  
            },  
            "eventTime": "2018-03-26T21:46:22Z",  
            "eventSource": "rekognition.amazonaws.com",  
            "eventName": "CreateCollection",  
            "awsRegion": "us-east-1",  
            "sourceIPAddress": "127.0.0.1",  
            "userAgent": "aws-internal/3",  
            "requestParameters": {  
                "collectionId": "8fa6aa65-cab4-4d0a-b976-7fd5df63f38a"  
            },  
            "responseElements": {  
                "collectionArn": "awsrekognition:us-  
east-1:111122223333:collection/8fa6aa65-cab4-4d0a-b976-7fd5df63f38a",  
                "faceModelVersion": "2.0",  
                "statusCode": 200  
            },  
            "requestID": "1e77d2d5-313f-11e8-8c0e-75c0272f31a4",  
            "eventID": "c6da4992-a9a1-4962-93b6-7d0483d95c30",  
            "eventType": "AwsApiCall",  
            "recipientAccountId": null  
        }  
    ]  
}
```

```
        "recipientAccountId": "111122223333"
    },
    {
        "eventVersion": "1.05",
        "userIdentity": {
            "type": "IAMUser",
            "principalId": "EX_PRINCIPAL_ID",
            "arn": "arn:aws:iam::111122223333:user/Alice",
            "accountId": "111122223333",
            "accessKeyId": "EXAMPLE_KEY_ID",
            "userName": "Alice"
        },
        "eventTime": "2018-03-26T21:46:25Z",
        "eventSource": "rekognition.amazonaws.com",
        "eventName": "DeleteCollection",
        "awsRegion": "us-east-1",
        "sourceIPAddress": "127.0.0.1",
        "userAgent": "aws-internal/3",
        "requestParameters": {
            "collectionId": "8fa6aa65-cab4-4d0a-b976-7fd5df63f38a"
        },
        "responseElements": {
            "statusCode": 200
        },
        "requestID": "213d5b78-313f-11e8-8c0e-75c0272f31a4",
        "eventID": "3ed4f4c9-22f8-4de4-a051-0d9d0c2faec9",
        "eventType": "AwsApiCall",
        "recipientAccountId": "111122223333"
    },
    {
        "eventVersion": "1.05",
        "userIdentity": {
            "type": "AssumedRole",
            "principalId": "EX_PRINCIPAL_ID",
            "arn": "arn:aws:sts::111122223333:assumed-role/Admin/Alice",
            "accountId": "111122223333",
            "accessKeyId": "EXAMPLE_KEY_ID",
            "sessionContext": {
                "attributes": {
                    "mfaAuthenticated": "false",
                    "creationDate": "2018-03-26T21:48:49Z"
                },
                "sessionIssuer": {
                    "type": "Role",
                    "principalId": "EX_PRINCIPAL_ID",
                    "arn": "arn:aws:iam::111122223333:role/Admin",
                    "accountId": "111122223333",
                    "userName": "Admin"
                }
            }
        },
        "eventTime": "2018-03-26T21:53:09Z",
        "eventSource": "rekognition.amazonaws.com",
        "eventName": "ListCollections",
        "awsRegion": "us-east-1",
        "sourceIPAddress": "127.0.0.1",
        "userAgent": "aws-cli/1.14.63 Python/3.4.7
Linux/3.2.45-0.6.wd.971.49.326.metal1.x86_64 botocore/1.9.16",
        "requestParameters": null,
        "responseElements": null,
        "requestID": "116a57f5-3140-11e8-8c0e-75c0272f31a4",
        "eventID": "94bb5ddd-7836-4fb1-a63e-a782eb009824",
        "eventType": "AwsApiCall",
        "recipientAccountId": "111122223333"
    },
    {
```

```
"eventVersion": "1.05",
"userIdentity": {
    "type": "AssumedRole",
    "principalId": "EX_PRINCIPAL_ID",
    "arn": "arn:aws:sts::111122223333:assumed-role/Admin/Alice",
    "accountId": "111122223333",
    "accessKeyId": "EXAMPLE_KEY_ID",
    "sessionContext": {
        "attributes": {
            "mfaAuthenticated": "false",
            "creationDate": "2018-03-26T21:48:49Z"
        },
        "sessionIssuer": {
            "type": "Role",
            "principalId": "EX_PRINCIPAL_ID",
            "arn": "arn:aws:iam::111122223333:role/Admin",
            "accountId": "111122223333",
            "userName": "Admin"
        }
    }
},
"eventTime": "2018-03-26T22:06:19Z",
"eventSource": "rekognition.amazonaws.com",
"eventName": "CreateStreamProcessor",
"awsRegion": "us-east-1",
"sourceIPAddress": "127.0.0.1",
"userAgent": "aws-cli/1.14.63 Python/3.4.7
Linux/3.2.45-0.6.wd.971.49.326.metall.x86_64 botocore/1.9.16",
"requestParameters": {
    "roleArn": "arn:aws:iam::111122223333:role/AmazonRekognition-
StreamProcessorRole",
    "settings": {
        "faceSearch": {
            "collectionId": "test"
        }
    },
    "name": "ProcessorName",
    "input": {
        "kinesisVideoStream": {
            "arn": "arn:aws:kinesisvideo:us-east-1:111122223333:stream/
VideoStream"
        }
    },
    "output": {
        "kinesisDataStream": {
            "arn": "arn:aws:kinesis:us-east-1:111122223333:stream/
AmazonRekognition-DataStream"
        }
    }
},
"responseElements": {
    "StreamProcessorArn": "arn:aws:rekognition:us-
east-1:111122223333:streamprocessor/ProcessorName"
},
"requestID": "e8fb2b3c-3141-11e8-8c0e-75c0272f31a4",
"eventID": "44ff8f90-fcc2-4740-9e57-0c47610df8e3",
"eventType": "AwsApiCall",
"recipientAccountId": "111122223333"
},
{
    "eventVersion": "1.05",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "EX_PRINCIPAL_ID",
        "arn": "arn:aws:sts::111122223333:assumed-role/Admin/Alice",
        "accountId": "111122223333",
```

```
"accessKeyId": "EXAMPLE_KEY_ID",
"sessionContext": {
    "attributes": {
        "mfaAuthenticated": "false",
        "creationDate": "2018-03-26T21:48:49Z"
    },
    "sessionIssuer": {
        "type": "Role",
        "principalId": "EX_PRINCIPAL_ID",
        "arn": "arn:aws:iam::111122223333:role/Admin",
        "accountId": "111122223333",
        "userName": "Admin"
    }
},
"eventTime": "2018-03-26T22:09:42Z",
"eventSource": "rekognition.amazonaws.com",
"eventName": "DeleteStreamProcessor",
"awsRegion": "us-east-1",
"sourceIPAddress": "127.0.0.1",
"userAgent": "aws-cli/1.14.63 Python/3.4.7
Linux/3.2.45-0.6.wd.971.49.326.metal1.x86_64 botocore/1.9.16",
"requestParameters": {
    "name": "ProcessorName"
},
"responseElements": null,
"requestID": "624c4c3e-3142-11e8-8c0e-75c0272f31a4",
"eventID": "27fd784e-fbf3-4163-9f0b-0006c6eed39f",
"eventType": "AwsApiCall",
"recipientAccountId": "111122223333"
},
{
    "eventVersion": "1.05",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "EX_PRINCIPAL_ID",
        "arn": "arn:aws:sts::111122223333:assumed-role/Admin/Alice",
        "accountId": "111122223333",
        "accessKeyId": "EXAMPLE_KEY_ID",
        "sessionContext": {
            "attributes": {
                "mfaAuthenticated": "false",
                "creationDate": "2018-03-26T21:48:49Z"
            },
            "sessionIssuer": {
                "type": "Role",
                "principalId": "EX_PRINCIPAL_ID",
                "arn": "arn:aws:iam::111122223333:role/Admin",
                "accountId": "111122223333",
                "userName": "Admin"
            }
        }
    },
    "eventTime": "2018-03-26T21:56:14Z",
    "eventSource": "rekognition.amazonaws.com",
    "eventName": "ListStreamProcessors",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "127.0.0.1",
    "userAgent": "aws-cli/1.14.63 Python/3.4.7
Linux/3.2.45-0.6.wd.971.49.326.metal1.x86_64 botocore/1.9.16",
    "requestParameters": null,
    "responseElements": null,
    "requestID": "811735f9-3140-11e8-8c0e-75c0272f31a4",
    "eventID": "5cfcc86a6-758c-4fb9-af13-557b04805c4e",
    "eventType": "AwsApiCall",
    "recipientAccountId": "111122223333"
```

```
        },
        {
            "eventVersion": "1.05",
            "userIdentity": {
                "type": "AssumedRole",
                "principalId": "EX_PRINCIPAL_ID",
                "arn": "arn:aws:sts::111122223333:assumed-role/Admin/Alice",
                "accountId": "111122223333",
                "accessKeyId": "EXAMPLE_KEY_ID",
                "sessionContext": {
                    "attributes": {
                        "mfaAuthenticated": "false",
                        "creationDate": "2018-03-26T22:55:22Z"
                    },
                    "sessionIssuer": {
                        "type": "Role",
                        "principalId": "EX_PRINCIPAL_ID",
                        "arn": "arn:aws:iam::111122223333:role/Admin",
                        "accountId": "111122223333",
                        "userName": "Admin"
                    }
                }
            },
            "eventTime": "2018-03-26T22:57:38Z",
            "eventSource": "rekognition.amazonaws.com",
            "eventName": "DescribeStreamProcessor",
            "awsRegion": "us-east-1",
            "sourceIPAddress": "127.0.0.1",
            "userAgent": "aws-cli/1.14.63 Python/3.4.7
Linux/3.2.45-0.6.wd.971.49.326.metal1.x86_64 botocore/1.9.16",
            "requestParameters": {
                "name": "ProcessorName"
            },
            "responseElements": null,
            "requestID": "14b2dd34-3149-11e8-8c0e-75c0272f31a4",
            "eventID": "0db3498c-e084-4b30-b5fb-aa0b71ef9b7b",
            "eventType": "AwsApiCall",
            "recipientAccountId": "111122223333"
        }
    ]
}
```

API Reference

이 단원에서는 Amazon Rekognition API 작업에 대한 설명서를 제공합니다.

HTTP 헤더

Amazon Rekognition HTTP 작업에는 일반적인 HTTP 헤더 외에 다음과 같은 필수 헤더가 있습니다.

헤더	값	설명
Content-Type:	application/x-amz-json-1.1	요청 콘텐츠가 JSON이라고 지정합니다. 또한 JSON 버전을 지정합니다.
X-Amz-Date:	<날짜>	Authorization 헤더에 저장되는 서명을 생성할 때 사용할 수 있는 날짜입니다. 형식은 'YYYYMMDD'T'HHMMSS'Z' 형식의 ISO 8601 기본이어야 합니다. 예를 들어 다음 날짜/시간 '20141123T120000Z'는 Amazon Rekognition에 사용할 수 있는 유효한 x-amz-date입니다.
X-Amz-Target:	RekognitionService.<작업>	대상 Amazon Rekognition 작업. 예를 들어 ListCollections 작업을 호출하는데 RekognitionService.ListCollections를 사용합니다.

항목

- [Actions \(p. 217\)](#)
- [Data Types \(p. 340\)](#)

Actions

The following actions are supported:

- [CompareFaces \(p. 219\)](#)
- [CreateCollection \(p. 224\)](#)
- [CreateStreamProcessor \(p. 227\)](#)
- [DeleteCollection \(p. 230\)](#)
- [DeleteFaces \(p. 232\)](#)
- [DeleteStreamProcessor \(p. 234\)](#)
- [DescribeCollection \(p. 236\)](#)
- [DescribeStreamProcessor \(p. 239\)](#)

- [DetectFaces \(p. 243\)](#)
- [DetectLabels \(p. 247\)](#)
- [DetectModerationLabels \(p. 251\)](#)
- [DetectText \(p. 254\)](#)
- [GetCelebrityInfo \(p. 257\)](#)
- [GetCelebrityRecognition \(p. 259\)](#)
- [GetContentModeration \(p. 264\)](#)
- [GetFaceDetection \(p. 268\)](#)
- [GetFaceSearch \(p. 273\)](#)
- [GetLabelDetection \(p. 278\)](#)
- [GetPersonTracking \(p. 282\)](#)
- [IndexFaces \(p. 287\)](#)
- [ListCollections \(p. 295\)](#)
- [ListFaces \(p. 298\)](#)
- [ListStreamProcessors \(p. 301\)](#)
- [RecognizeCelebrities \(p. 304\)](#)
- [SearchFaces \(p. 308\)](#)
- [SearchFacesByImage \(p. 311\)](#)
- [StartCelebrityRecognition \(p. 315\)](#)
- [StartContentModeration \(p. 318\)](#)
- [StartFaceDetection \(p. 322\)](#)
- [StartFaceSearch \(p. 326\)](#)
- [StartLabelDetection \(p. 330\)](#)
- [StartPersonTracking \(p. 334\)](#)
- [StartStreamProcessor \(p. 337\)](#)
- [StopStreamProcessor \(p. 339\)](#)

CompareFaces

Compares a face in the source input image with each of the 100 largest faces detected in the target input image.

Note

If the source image contains multiple faces, the service detects the largest face and compares it with each face detected in the target image.

You pass the input and target images either as base64-encoded image bytes or as references to images in an Amazon S3 bucket. If you use the AWS CLI to call Amazon Rekognition operations, passing image bytes isn't supported. The image must be formatted as a PNG or JPEG file.

In response, the operation returns an array of face matches ordered by similarity score in descending order. For each face match, the response provides a bounding box of the face, facial landmarks, pose details (pitch, role, and yaw), quality (brightness and sharpness), and confidence value (indicating the level of confidence that the bounding box contains a face). The response also provides a similarity score, which indicates how closely the faces match.

Note

By default, only faces with a similarity score of greater than or equal to 80% are returned in the response. You can change this value by specifying the `SimilarityThreshold` parameter.

`CompareFaces` also returns an array of faces that don't match the source image. For each face, it returns a bounding box, confidence value, landmarks, pose details, and quality. The response also returns information about the face in the source image, including the bounding box of the face and confidence value.

If the image doesn't contain Exif metadata, `CompareFaces` returns orientation information for the source and target images. Use these values to display the images with the correct image orientation.

If no faces are detected in the source or target images, `CompareFaces` returns an `InvalidParameterException` error.

Note

This is a stateless API operation. That is, data returned by this operation doesn't persist.

For an example, see [이미지에 있는 얼굴 비교 \(p. 114\)](#).

This operation requires permissions to perform the `rekognition:CompareFaces` action.

Request Syntax

```
{  
    "SimilarityThreshold": number,  
    "SourceImage": {  
        "Bytes": blob,  
        "S3Object": {  
            "Bucket": "string",  
            "Name": "string",  
            "Version": "string"  
        }  
    },  
    "TargetImage": {  
        "Bytes": blob,  
        "S3Object": {  
            "Bucket": "string",  
            "Name": "string",  
            "Version": "string"  
        }  
    }  
}
```

```
        "Name": "string",
        "Version": "string"
    }
}
```

Request Parameters

The request accepts the following data in JSON format.

[SimilarityThreshold \(p. 219\)](#)

The minimum level of confidence in the face matches that a match must meet to be included in the `FaceMatches` array.

Type: Float

Valid Range: Minimum value of 0. Maximum value of 100.

Required: No

[SourceImage \(p. 219\)](#)

The input image as base64-encoded bytes or an S3 object. If you use the AWS CLI to call Amazon Rekognition operations, passing base64-encoded image bytes is not supported.

Type: [Image \(p. 368\)](#) object

Required: Yes

[TargetImage \(p. 219\)](#)

The target image as base64-encoded bytes or an S3 object. If you use the AWS CLI to call Amazon Rekognition operations, passing base64-encoded image bytes is not supported.

Type: [Image \(p. 368\)](#) object

Required: Yes

Response Syntax

```
{
    "FaceMatches": [
        {
            "Face": {
                "BoundingBox": {
                    "Height": number,
                    "Left": number,
                    "Top": number,
                    "Width": number
                },
                "Confidence": number,
                "Landmarks": [
                    {
                        "Type": "string",
                        "X": number,
                        "Y": number
                    }
                ],
                "Pose": {
                    "Pitch": number,

```

```
        "Roll": number,
        "Yaw": number
    },
    "Quality": {
        "Brightness": number,
        "Sharpness": number
    }
},
"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,
    "Landmarks": [
        {
            "Type": "string",
            "X": number,
            "Y": number
        }
    ],
    "Pose": {
        "Pitch": number,
        "Roll": number,
        "Yaw": number
    },
    "Quality": {
        "Brightness": number,
        "Sharpness": number
    }
}
]
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

FaceMatches (p. 220)

An array of faces in the target image that match the source image face. Each `CompareFacesMatch` object provides the bounding box, the confidence level that the bounding box contains a face, and the similarity score for the face in the bounding box and the face in the source image.

Type: Array of `CompareFacesMatch` (p. 352) objects

[SourceImageFace \(p. 220\)](#)

The face in the source image that was used for comparison.

Type: [ComparedSourceImageFace \(p. 351\)](#) object

[SourceImageOrientationCorrection \(p. 220\)](#)

The value of `SourceImageOrientationCorrection` is always null.

If the input image is in .jpeg format, it might contain exchangeable image file format (Exif) metadata that includes the image's orientation. Amazon Rekognition uses this orientation information to perform image correction. The bounding box coordinates are translated to represent object locations after the orientation information in the Exif metadata is used to correct the image orientation. Images in .png format don't contain Exif metadata.

Amazon Rekognition doesn't perform image correction for images in .png format and .jpeg images without orientation information in the image Exif metadata. The bounding box coordinates aren't translated and represent the object locations before the image is rotated.

Type: String

Valid Values: `ROTATE_0` | `ROTATE_90` | `ROTATE_180` | `ROTATE_270`

[TargetImageOrientationCorrection \(p. 220\)](#)

The value of `TargetImageOrientationCorrection` is always null.

If the input image is in .jpeg format, it might contain exchangeable image file format (Exif) metadata that includes the image's orientation. Amazon Rekognition uses this orientation information to perform image correction. The bounding box coordinates are translated to represent object locations after the orientation information in the Exif metadata is used to correct the image orientation. Images in .png format don't contain Exif metadata.

Amazon Rekognition doesn't perform image correction for images in .png format and .jpeg images without orientation information in the image Exif metadata. The bounding box coordinates aren't translated and represent the object locations before the image is rotated.

Type: String

Valid Values: `ROTATE_0` | `ROTATE_90` | `ROTATE_180` | `ROTATE_270`

[UnmatchedFaces \(p. 220\)](#)

An array of faces in the target image that did not match the source image face.

Type: Array of [ComparedFace \(p. 350\)](#) objects

Errors

AccessDeniedException

You are not authorized to perform the action.

HTTP Status Code: 400

ImageTooLargeException

The input image size exceeds the allowed limit. For more information, see [Amazon Rekognition의 제한 \(p. 400\)](#).

HTTP Status Code: 400

InternalServerError

Amazon Rekognition experienced a service issue. Try your call again.

HTTP Status Code: 500

InvalidImageFormatException

The provided image format is not supported.

HTTP Status Code: 400

InvalidParameterException

Input parameter violated a constraint. Validate your parameter before calling the API operation again.

HTTP Status Code: 400

InvalidS3ObjectException

Amazon Rekognition is unable to access the S3 object specified in the request.

HTTP Status Code: 400

ProvisionedThroughputExceededException

The number of requests exceeded your throughput limit. If you want to increase this limit, contact Amazon Rekognition.

HTTP Status Code: 400

ThrottlingException

Amazon Rekognition is temporarily unable to process the request. Try your call again.

HTTP Status Code: 500

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

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

CreateCollection

Creates a collection in an AWS Region. You can add faces to the collection using the [IndexFaces \(p. 287\)](#) operation.

For example, you might create collections, one for each of your application users. A user can then index faces using the `IndexFaces` operation and persist results in a specific collection. Then, a user can search the collection for faces in the user-specific container.

When you create a collection, it is associated with the latest version of the face model version.

Note

Collection names are case-sensitive.

This operation requires permissions to perform the `rekognition:CreateCollection` action.

Request Syntax

```
{  
    "CollectionId": "string"  
}
```

Request Parameters

The request accepts the following data in JSON format.

[CollectionId \(p. 224\)](#)

ID for the collection that you are creating.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 255.

Pattern: [a-zA-Z0-9_.\-\]+

Required: Yes

Response Syntax

```
{  
    "CollectionArn": "string",  
    "FaceModelVersion": "string",  
    "StatusCode": number  
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

[CollectionArn \(p. 224\)](#)

Amazon Resource Name (ARN) of the collection. You can use this to manage permissions on your resources.

Type: String

[FaceModelVersion \(p. 224\)](#)

Version number of the face detection model associated with the collection you are creating.

Type: String

[StatusCode \(p. 224\)](#)

HTTP status code indicating the result of the operation.

Type: Integer

Valid Range: Minimum value of 0.

Errors

[AccessDeniedException](#)

You are not authorized to perform the action.

HTTP Status Code: 400

[InternalServerError](#)

Amazon Rekognition experienced a service issue. Try your call again.

HTTP Status Code: 500

[InvalidParameterException](#)

Input parameter violated a constraint. Validate your parameter before calling the API operation again.

HTTP Status Code: 400

[ProvisionedThroughputExceededException](#)

The number of requests exceeded your throughput limit. If you want to increase this limit, contact Amazon Rekognition.

HTTP Status Code: 400

[ResourceAlreadyExistsException](#)

A collection with the specified ID already exists.

HTTP Status Code: 400

[ThrottlingException](#)

Amazon Rekognition is temporarily unable to process the request. Try your call again.

HTTP Status Code: 500

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)

- AWS SDK for Java
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V2

CreateStreamProcessor

Creates an Amazon Rekognition stream processor that you can use to detect and recognize faces in a streaming video.

Amazon Rekognition Video is a consumer of live video from Amazon Kinesis Video Streams. Amazon Rekognition Video sends analysis results to Amazon Kinesis Data Streams.

You provide as input a Kinesis video stream (`Input`) and a Kinesis data stream (`Output`) stream. You also specify the face recognition criteria in `Settings`. For example, the collection containing faces that you want to recognize. Use `Name` to assign an identifier for the stream processor. You use `Name` to manage the stream processor. For example, you can start processing the source video by calling [StartStreamProcessor \(p. 337\)](#) with the `Name` field.

After you have finished analyzing a streaming video, use [StopStreamProcessor \(p. 339\)](#) to stop processing. You can delete the stream processor by calling [DeleteStreamProcessor \(p. 234\)](#).

Request Syntax

```
{  
    "Input": {  
        "KinesisVideoStream": {  
            "Arn": "string"  
        }  
    },  
    "Name": "string",  
    "Output": {  
        "KinesisDataStream": {  
            "Arn": "string"  
        }  
    },  
    "RoleArn": "string",  
    "Settings": {  
        "FaceSearch": {  
            "CollectionId": "string",  
            "FaceMatchThreshold": number  
        }  
    }  
}
```

Request Parameters

The request accepts the following data in JSON format.

[Input \(p. 227\)](#)

Kinesis video stream stream that provides the source streaming video. If you are using the AWS CLI, the parameter name is `StreamProcessorInput`.

Type: [StreamProcessorInput \(p. 390\)](#) object

Required: Yes

[Name \(p. 227\)](#)

An identifier you assign to the stream processor. You can use `Name` to manage the stream processor. For example, you can get the current status of the stream processor by calling [DescribeStreamProcessor \(p. 239\)](#). `Name` is idempotent.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 128.

Pattern: [a-zA-Z0-9_.\-_]+

Required: Yes

[Output \(p. 227\)](#)

Kinesis data stream stream to which Amazon Rekognition Video puts the analysis results. If you are using the AWS CLI, the parameter name is StreamProcessorOutput.

Type: [StreamProcessorOutput \(p. 391\)](#) object

Required: Yes

[RoleArn \(p. 227\)](#)

ARN of the IAM role that allows access to the stream processor.

Type: String

Pattern: arn:aws:iam::\d{12}:role/?[a-zA-Z_0-9+=,.@\-_/\-]+

Required: Yes

[Settings \(p. 227\)](#)

Face recognition input parameters to be used by the stream processor. Includes the collection to use for face recognition and the face attributes to detect.

Type: [StreamProcessorSettings \(p. 392\)](#) object

Required: Yes

Response Syntax

```
{  
    "StreamProcessorArn": "string"  
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

[StreamProcessorArn \(p. 228\)](#)

ARN for the newly create stream processor.

Type: String

Pattern: (^arn:[a-zA-Z\d-]+:rekognition:[a-zA-Z\d-]+:\d{12}:streamprocessor\/.+\\$)

Errors

AccessDeniedException

You are not authorized to perform the action.

HTTP Status Code: 400
InternalServerError

Amazon Rekognition experienced a service issue. Try your call again.

HTTP Status Code: 500
InvalidParameterException

Input parameter violated a constraint. Validate your parameter before calling the API operation again.

HTTP Status Code: 400
LimitExceededException

An Amazon Rekognition service limit was exceeded. For example, if you start too many Amazon Rekognition Video jobs concurrently, calls to start operations (`StartLabelDetection`, for example) will raise a `LimitExceededException` exception (HTTP status code: 400) until the number of concurrently running jobs is below the Amazon Rekognition service limit.

HTTP Status Code: 400
ProvisionedThroughputExceededException

The number of requests exceeded your throughput limit. If you want to increase this limit, contact Amazon Rekognition.

HTTP Status Code: 400
ResourceInUseException

HTTP Status Code: 400
ThrottlingException

Amazon Rekognition is temporarily unable to process the request. Try your call again.

HTTP Status Code: 500

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

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

DeleteCollection

Deletes the specified collection. Note that this operation removes all faces in the collection. For an example, see [모음 삭제](#) (p. 134).

This operation requires permissions to perform the `rekognition:DeleteCollection` action.

Request Syntax

```
{  
    "CollectionId": "string"  
}
```

Request Parameters

The request accepts the following data in JSON format.

[CollectionId](#) (p. 230)

ID of the collection to delete.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 255.

Pattern: [a-zA-Z0-9_.\-\-]+

Required: Yes

Response Syntax

```
{  
    "StatusCode": number  
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

[StatusCode](#) (p. 230)

HTTP status code that indicates the result of the operation.

Type: Integer

Valid Range: Minimum value of 0.

Errors

[AccessDeniedException](#)

You are not authorized to perform the action.

HTTP Status Code: 400
InternalServerError

Amazon Rekognition experienced a service issue. Try your call again.

HTTP Status Code: 500
InvalidParameterException

Input parameter violated a constraint. Validate your parameter before calling the API operation again.

HTTP Status Code: 400
ProvisionedThroughputExceededException

The number of requests exceeded your throughput limit. If you want to increase this limit, contact Amazon Rekognition.

HTTP Status Code: 400
ResourceNotFoundException

The collection specified in the request cannot be found.

HTTP Status Code: 400
ThrottlingException

Amazon Rekognition is temporarily unable to process the request. Try your call again.

HTTP Status Code: 500

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

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

DeleteFaces

Deletes faces from a collection. You specify a collection ID and an array of face IDs to remove from the collection.

This operation requires permissions to perform the `rekognition:DeleteFaces` action.

Request Syntax

```
{  
    "CollectionId": "string",  
    "FaceIds": [ "string" ]  
}
```

Request Parameters

The request accepts the following data in JSON format.

[CollectionId \(p. 232\)](#)

Collection from which to remove the specific faces.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 255.

Pattern: [a-zA-Z0-9_.\-_]+

Required: Yes

[Facelids \(p. 232\)](#)

An array of face IDs to delete.

Type: Array of strings

Array Members: Minimum number of 1 item. Maximum number of 4096 items.

Pattern: [0-9a-f]{8}-[0-9a-f]{4}-[0-9a-f]{4}-[0-9a-f]{4}-[0-9a-f]{12}

Required: Yes

Response Syntax

```
{  
    "DeletedFaces": [ "string" ]  
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

[DeletedFaces \(p. 232\)](#)

An array of strings (face IDs) of the faces that were deleted.

Type: Array of strings

Array Members: Minimum number of 1 item. Maximum number of 4096 items.

Pattern: [0-9a-f]{8}-[0-9a-f]{4}-[0-9a-f]{4}-[0-9a-f]{4}-[0-9a-f]{12}

Errors

AccessDeniedException

You are not authorized to perform the action.

HTTP Status Code: 400

InternalServerError

Amazon Rekognition experienced a service issue. Try your call again.

HTTP Status Code: 500

InvalidParameterException

Input parameter violated a constraint. Validate your parameter before calling the API operation again.

HTTP Status Code: 400

ProvisionedThroughputExceededException

The number of requests exceeded your throughput limit. If you want to increase this limit, contact Amazon Rekognition.

HTTP Status Code: 400

ResourceNotFoundException

The collection specified in the request cannot be found.

HTTP Status Code: 400

ThrottlingException

Amazon Rekognition is temporarily unable to process the request. Try your call again.

HTTP Status Code: 500

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

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

DeleteStreamProcessor

Deletes the stream processor identified by `Name`. You assign the value for `Name` when you create the stream processor with [CreateStreamProcessor \(p. 227\)](#). You might not be able to use the same name for a stream processor for a few seconds after calling `DeleteStreamProcessor`.

Request Syntax

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

Request Parameters

The request accepts the following data in JSON format.

Name (p. 234)

The name of the stream processor you want to delete.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 128.

Pattern: [a-zA-Z0-9_.\-\-]+

Required: Yes

Response Elements

If the action is successful, the service sends back an HTTP 200 response with an empty HTTP body.

Errors

AccessDeniedException

You are not authorized to perform the action.

HTTP Status Code: 400

InternalServerError

Amazon Rekognition experienced a service issue. Try your call again.

HTTP Status Code: 500

InvalidParameterException

Input parameter violated a constraint. Validate your parameter before calling the API operation again.

HTTP Status Code: 400

ProvisionedThroughputExceededException

The number of requests exceeded your throughput limit. If you want to increase this limit, contact Amazon Rekognition.

HTTP Status Code: 400

ResourceInUseException

HTTP Status Code: 400

ResourceNotFoundException

The collection specified in the request cannot be found.

HTTP Status Code: 400

ThrottlingException

Amazon Rekognition is temporarily unable to process the request. Try your call again.

HTTP Status Code: 500

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

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

DescribeCollection

Describes the specified collection. You can use `DescribeCollection` to get information, such as the number of faces indexed into a collection and the version of the model used by the collection for face detection.

For more information, see [???](#).

Request Syntax

```
{  
    "CollectionId": "string"  
}
```

Request Parameters

The request accepts the following data in JSON format.

[CollectionId \(p. 236\)](#)

The ID of the collection to describe.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 255.

Pattern: [a-zA-Z0-9_.\-\]+

Required: Yes

Response Syntax

```
{  
    "CollectionARN": "string",  
    "CreationTimestamp": number,  
    "FaceCount": number,  
    "FaceModelVersion": "string"  
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

[CollectionARN \(p. 236\)](#)

The Amazon Resource Name (ARN) of the collection.

Type: String

[CreationTimestamp \(p. 236\)](#)

The number of milliseconds since the Unix epoch time until the creation of the collection. The Unix epoch time is 00:00:00 Coordinated Universal Time (UTC), Thursday, 1 January 1970.

Type: Timestamp

[FaceCount \(p. 236\)](#)

The number of faces that are indexed into the collection. To index faces into a collection, use [IndexFaces \(p. 287\)](#).

Type: Long

Valid Range: Minimum value of 0.

[FaceModelVersion \(p. 236\)](#)

The version of the face model that's used by the collection for face detection.

For more information, see [모델 버전 관리 \(p. 9\)](#).

Type: String

Errors

AccessDeniedException

You are not authorized to perform the action.

HTTP Status Code: 400

InternalServerError

Amazon Rekognition experienced a service issue. Try your call again.

HTTP Status Code: 500

InvalidParameterException

Input parameter violated a constraint. Validate your parameter before calling the API operation again.

HTTP Status Code: 400

ProvisionedThroughputExceededException

The number of requests exceeded your throughput limit. If you want to increase this limit, contact Amazon Rekognition.

HTTP Status Code: 400

ResourceNotFoundException

The collection specified in the request cannot be found.

HTTP Status Code: 400

ThrottlingException

Amazon Rekognition is temporarily unable to process the request. Try your call again.

HTTP Status Code: 500

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)

- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V2

DescribeStreamProcessor

Provides information about a stream processor created by [CreateStreamProcessor \(p. 227\)](#). You can get information about the input and output streams, the input parameters for the face recognition being performed, and the current status of the stream processor.

Request Syntax

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

Request Parameters

The request accepts the following data in JSON format.

Name (p. 239)

Name of the stream processor for which you want information.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 128.

Pattern: [a-zA-Z0-9_.\-\-]+

Required: Yes

Response Syntax

```
{  
    "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"  
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

[CreationTimestamp \(p. 239\)](#)

Date and time the stream processor was created

Type: Timestamp

[Input \(p. 239\)](#)

Kinesis video stream that provides the source streaming video.

Type: [StreamProcessorInput \(p. 390\)](#) object

[LastUpdateTimestamp \(p. 239\)](#)

The time, in Unix format, the stream processor was last updated. For example, when the stream processor moves from a running state to a failed state, or when the user starts or stops the stream processor.

Type: Timestamp

[Name \(p. 239\)](#)

Name of the stream processor.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 128.

Pattern: [a-zA-Z0-9_.\-\-]+

[Output \(p. 239\)](#)

Kinesis data stream to which Amazon Rekognition Video puts the analysis results.

Type: [StreamProcessorOutput \(p. 391\)](#) object

[RoleArn \(p. 239\)](#)

ARN of the IAM role that allows access to the stream processor.

Type: String

Pattern: arn:aws:iam::\d{12}:role/?[a-zA-Z_0-9+=,.@\-_/\-]+

[Settings \(p. 239\)](#)

Face recognition input parameters that are being used by the stream processor. Includes the collection to use for face recognition and the face attributes to detect.

Type: [StreamProcessorSettings \(p. 392\)](#) object

[Status \(p. 239\)](#)

Current status of the stream processor.

Type: String

Valid Values: STOPPED | STARTING | RUNNING | FAILED | STOPPING

[StatusMessage \(p. 239\)](#)

Detailed status message about the stream processor.

Type: String

[StreamProcessorArn \(p. 239\)](#)

ARN of the stream processor.

Type: String

Pattern: (^arn:[a-z\d-]+:rekognition:[a-z\d-]+\:\d{12}:streamprocessor\/.+)\$)

Errors

AccessDeniedException

You are not authorized to perform the action.

HTTP Status Code: 400

InternalServerError

Amazon Rekognition experienced a service issue. Try your call again.

HTTP Status Code: 500

InvalidParameterException

Input parameter violated a constraint. Validate your parameter before calling the API operation again.

HTTP Status Code: 400

ProvisionedThroughputExceededException

The number of requests exceeded your throughput limit. If you want to increase this limit, contact Amazon Rekognition.

HTTP Status Code: 400

ResourceNotFoundException

The collection specified in the request cannot be found.

HTTP Status Code: 400

ThrottlingException

Amazon Rekognition is temporarily unable to process the request. Try your call again.

HTTP Status Code: 500

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

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

DetectFaces

Detects faces within an image that is provided as input.

`DetectFaces` detects the 100 largest faces in the image. For each face detected, the operation returns face details. These details include a bounding box of the face, a confidence value (that the bounding box contains a face), and a fixed set of attributes such as facial landmarks (for example, coordinates of eye and mouth), gender, presence of beard, sunglasses, and so on.

The face-detection algorithm is most effective on frontal faces. For non-frontal or obscured faces, the algorithm might not detect the faces or might detect faces with lower confidence.

You pass the input image either as base64-encoded image bytes or as a reference to an image in an Amazon S3 bucket. If you use the `Image` parameter to call Amazon Rekognition operations, passing image bytes is not supported. The image must be either a PNG or JPEG formatted file.

Note

This is a stateless API operation. That is, the operation does not persist any data.

This operation requires permissions to perform the `rekognition:DetectFaces` action.

Request Syntax

```
{  
    "Attributes": [ "string" ],  
    "Image": {  
        "Bytes": blob,  
        "S3Object": {  
            "Bucket": "string",  
            "Name": "string",  
            "Version": "string"  
        }  
    }  
}
```

Request Parameters

The request accepts the following data in JSON format.

Attributes (p. 243)

An array of facial attributes you want to be returned. This can be the default list of attributes or all attributes. If you don't specify a value for `Attributes` or if you specify `["DEFAULT"]`, the API returns the following subset of facial attributes: `BoundingBox`, `Confidence`, `Pose`, `Quality`, and `Landmarks`. If you provide `["ALL"]`, all facial attributes are returned, but the operation takes longer to complete.

If you provide both, `["ALL" , "DEFAULT"]`, the service uses a logical AND operator to determine which attributes to return (in this case, all attributes).

Type: Array of strings

Valid Values: `DEFAULT` | `ALL`

Required: No

Image (p. 243)

The input image as base64-encoded bytes or an S3 object. If you use the AWS CLI to call Amazon Rekognition operations, passing base64-encoded image bytes is not supported.

Type: [Image \(p. 368\)](#) object

Required: Yes

Response Syntax

```
{  
    "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"  
                }  
            ],  
            "Eglasses": {  
                "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": {  
                "Angle": number,  
                "Type": "string"  
            }  
        }  
    ]  
}
```

```
    "Pitch": number,
    "Roll": number,
    "Yaw": number
},
"Quality": {
    "Brightness": number,
    "Sharpness": number
},
"Smile": {
    "Confidence": number,
    "Value": boolean
},
"Sunglasses": {
    "Confidence": number,
    "Value": boolean
}
},
"OrientationCorrection": "string"
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

[FaceDetails \(p. 244\)](#)

Details of each face found in the image.

Type: Array of [FaceDetail \(p. 359\)](#) objects

[OrientationCorrection \(p. 244\)](#)

The value of OrientationCorrection is always null.

If the input image is in .jpeg format, it might contain exchangeable image file format (Exif) metadata that includes the image's orientation. Amazon Rekognition uses this orientation information to perform image correction. The bounding box coordinates are translated to represent object locations after the orientation information in the Exif metadata is used to correct the image orientation. Images in .png format don't contain Exif metadata.

Amazon Rekognition doesn't perform image correction for images in .png format and .jpeg images without orientation information in the image Exif metadata. The bounding box coordinates aren't translated and represent the object locations before the image is rotated.

Type: String

Valid Values: ROTATE_0 | ROTATE_90 | ROTATE_180 | ROTATE_270

Errors

[AccessDeniedException](#)

You are not authorized to perform the action.

HTTP Status Code: 400

[ImageTooLargeException](#)

The input image size exceeds the allowed limit. For more information, see [Amazon Rekognition의 제한 \(p. 400\)](#).

HTTP Status Code: 400
InternalServerError

Amazon Rekognition experienced a service issue. Try your call again.

HTTP Status Code: 500
InvalidImageFormatException

The provided image format is not supported.

HTTP Status Code: 400
InvalidParameterException

Input parameter violated a constraint. Validate your parameter before calling the API operation again.

HTTP Status Code: 400
InvalidS3ObjectException

Amazon Rekognition is unable to access the S3 object specified in the request.

HTTP Status Code: 400
ProvisionedThroughputExceededException

The number of requests exceeded your throughput limit. If you want to increase this limit, contact Amazon Rekognition.

HTTP Status Code: 400
ThrottlingException

Amazon Rekognition is temporarily unable to process the request. Try your call again.

HTTP Status Code: 500

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

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

DetectLabels

Detects instances of real-world entities within an image (JPEG or PNG) provided as input. This includes objects like flower, tree, and table; events like wedding, graduation, and birthday party; and concepts like landscape, evening, and nature.

For an example, see [Amazon S3 버킷에 저장된 이미지 분석 \(p. 37\)](#).

Note

`DetectLabels` does not support the detection of activities. However, activity detection is supported for label detection in videos. For more information, see [StartLabelDetection \(p. 330\)](#).

You pass the input image as base64-encoded image bytes or as a reference to an image in an Amazon S3 bucket. If you use the AWS CLI to call Amazon Rekognition operations, passing image bytes is not supported. The image must be either a PNG or JPEG formatted file.

For each object, scene, and concept the API returns one or more labels. Each label provides the object name, and the level of confidence that the image contains the object. For example, suppose the input image has a lighthouse, the sea, and a rock. The response includes all three labels, one for each object.

```
{Name: lighthouse, Confidence: 98.4629}  
  
{Name: rock, Confidence: 79.2097}  
  
{Name: sea, Confidence: 75.061}
```

In the preceding example, the operation returns one label for each of the three objects. The operation can also return multiple labels for the same object in the image. For example, if the input image shows a flower (for example, a tulip), the operation might return the following three labels.

```
{Name: flower, Confidence: 99.0562}  
  
{Name: plant, Confidence: 99.0562}  
  
{Name: tulip, Confidence: 99.0562}
```

In this example, the detection algorithm more precisely identifies the flower as a tulip.

In response, the API returns an array of labels. In addition, the response also includes the orientation correction. Optionally, you can specify `MinConfidence` to control the confidence threshold for the labels returned. The default is 50%. You can also add the `MaxLabels` parameter to limit the number of labels returned.

Note

If the object detected is a person, the operation doesn't provide the same facial details that the [DetectFaces \(p. 243\)](#) operation provides.

`DetectLabels` returns bounding boxes for instances of common object labels in an array of [Instance \(p. 370\)](#) objects. An `Instance` object contains a [BoundingBox \(p. 344\)](#) object, for the location of the label on the image. It also includes the confidence by which the bounding box was detected.

`DetectLabels` also returns a hierarchical taxonomy of detected labels. For example, a detected car might be assigned the label car. The label car has two parent labels: Vehicle (its parent) and Transportation (its grandparent). The response returns the entire list of ancestors for a label. Each ancestor is a unique label in the response. In the previous example, Car, Vehicle, and Transportation are returned as unique labels in the response.

This is a stateless API operation. That is, the operation does not persist any data.

This operation requires permissions to perform the `rekognition:DetectLabels` action.

Request Syntax

```
{  
    "Image": {  
        "Bytes": blob,  
        "S3Object": {  
            "Bucket": "string",  
            "Name": "string",  
            "Version": "string"  
        }  
    },  
    "MaxLabels": number,  
    "MinConfidence": number  
}
```

Request Parameters

The request accepts the following data in JSON format.

[Image \(p. 248\)](#)

Type: [Image \(p. 368\)](#) object
The input image as base64-encoded bytes or an S3 object. If you use the AWS CLI to call Amazon Rekognition operations, passing base64-encoded image bytes is not supported.

Type: [Image \(p. 368\)](#) object

Required: Yes

[MaxLabels \(p. 248\)](#)

Maximum number of labels you want the service to return in the response. The service returns the specified number of highest confidence labels.

Type: Integer

Valid Range: Minimum value of 0.

Required: No

[MinConfidence \(p. 248\)](#)

Specifies the minimum confidence level for the labels to return. Amazon Rekognition doesn't return any labels with confidence lower than this specified value.

If [MinConfidence](#) is not specified, the operation returns labels with a confidence values greater than or equal to 50 percent.

Type: Float

Valid Range: Minimum value of 0. Maximum value of 100.

Required: No

Response Syntax

```
{  
    "LabelModelVersion": "string",  
    "Labels": [  
        {  
            "Confidence": number,  
            "Instances": [  
                {  
                    "Label": "string",  
                    "Confidence": number,  
                    "Geometry": {  
                        "BoundingBox": {  
                            "Width": number,  
                            "Height": number,  
                            "Left": number,  
                            "Top": number  
                        },  
                        "Orientation": number  
                    },  
                    "TextProperties": {  
                        "Text": "string",  
                        "Confidence": number  
                    }  
                }  
            ]  
        }  
    ]  
}
```

```
{  
    "BoundingBox": {  
        "Height": number,  
        "Left": number,  
        "Top": number,  
        "Width": number  
    },  
    "Confidence": number  
},  
]  
,"Name": "string",  
"Parents": [  
    {  
        "Name": "string"  
    }  
]  
],  
"OrientationCorrection": "string"  
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

[LabelModelVersion \(p. 248\)](#)

Version number of the label detection model that was used to detect labels.

Type: String

[Labels \(p. 248\)](#)

An array of labels for the real-world objects detected.

Type: Array of [Label \(p. 373\)](#) objects

[OrientationCorrection \(p. 248\)](#)

The value of `OrientationCorrection` is always null.

If the input image is in .jpeg format, it might contain exchangeable image file format (Exif) metadata that includes the image's orientation. Amazon Rekognition uses this orientation information to perform image correction. The bounding box coordinates are translated to represent object locations after the orientation information in the Exif metadata is used to correct the image orientation. Images in .png format don't contain Exif metadata.

Amazon Rekognition doesn't perform image correction for images in .png format and .jpeg images without orientation information in the image Exif metadata. The bounding box coordinates aren't translated and represent the object locations before the image is rotated.

Type: String

Valid Values: `ROTATE_0` | `ROTATE_90` | `ROTATE_180` | `ROTATE_270`

Errors

`AccessDeniedException`

You are not authorized to perform the action.

HTTP Status Code: 400

ImageTooLargeException

The input image size exceeds the allowed limit. For more information, see [Amazon Rekognition의 제한 \(p. 400\)](#).

HTTP Status Code: 400

InternalServerError

Amazon Rekognition experienced a service issue. Try your call again.

HTTP Status Code: 500

InvalidImageFormatException

The provided image format is not supported.

HTTP Status Code: 400

InvalidParameterException

Input parameter violated a constraint. Validate your parameter before calling the API operation again.

HTTP Status Code: 400

InvalidS3ObjectException

Amazon Rekognition is unable to access the S3 object specified in the request.

HTTP Status Code: 400

ProvisionedThroughputExceededException

The number of requests exceeded your throughput limit. If you want to increase this limit, contact Amazon Rekognition.

HTTP Status Code: 400

ThrottlingException

Amazon Rekognition is temporarily unable to process the request. Try your call again.

HTTP Status Code: 500

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

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

DetectModerationLabels

Detects explicit or suggestive adult content in a specified JPEG or PNG format image. Use `DetectModerationLabels` to moderate images depending on your requirements. For example, you might want to filter images that contain nudity, but not images containing suggestive content.

To filter images, use the labels returned by `DetectModerationLabels` to determine which types of content are appropriate.

For information about moderation labels, see [비안전 콘텐츠 감지 \(p. 187\)](#).

You pass the input image either as base64-encoded image bytes or as a reference to an image in an Amazon S3 bucket. If you use the AWS CLI to call Amazon Rekognition operations, passing image bytes is not supported. The image must be either a PNG or JPEG formatted file.

Request Syntax

```
{  
    "Image": {  
        "Bytes": blob,  
        "S3Object": {  
            "Bucket": "string",  
            "Name": "string",  
            "Version": "string"  
        }  
    },  
    "MinConfidence": number  
}
```

Request Parameters

The request accepts the following data in JSON format.

Image (p. 251)

The input image as base64-encoded bytes or an S3 object. If you use the AWS CLI to call Amazon Rekognition operations, passing base64-encoded image bytes is not supported.

Type: [Image \(p. 368\)](#) object

Required: Yes

MinConfidence (p. 251)

Specifies the minimum confidence level for the labels to return. Amazon Rekognition doesn't return any labels with a confidence level lower than this specified value.

If you don't specify `MinConfidence`, the operation returns labels with confidence values greater than or equal to 50 percent.

Type: Float

Valid Range: Minimum value of 0. Maximum value of 100.

Required: No

Response Syntax

```
{
```

```
"ModerationLabels": [  
    {  
        "Confidence": number,  
        "Name": "string",  
        "ParentName": "string"  
    }  
]
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

[ModerationLabels \(p. 251\)](#)

Array of detected Moderation labels and the time, in milliseconds from the start of the video, they were detected.

Type: Array of [ModerationLabel \(p. 377\)](#) objects

Errors

`AccessDeniedException`

You are not authorized to perform the action.

HTTP Status Code: 400

`ImageTooLargeException`

The input image size exceeds the allowed limit. For more information, see [Amazon Rekognition의 제한 \(p. 400\)](#).

HTTP Status Code: 400

`InternalServerError`

Amazon Rekognition experienced a service issue. Try your call again.

HTTP Status Code: 500

`InvalidImageFormatException`

The provided image format is not supported.

HTTP Status Code: 400

`InvalidParameterException`

Input parameter violated a constraint. Validate your parameter before calling the API operation again.

HTTP Status Code: 400

`InvalidS3ObjectException`

Amazon Rekognition is unable to access the S3 object specified in the request.

HTTP Status Code: 400

`ProvisionedThroughputExceededException`

The number of requests exceeded your throughput limit. If you want to increase this limit, contact Amazon Rekognition.

HTTP Status Code: 400
ThrottlingException

Amazon Rekognition is temporarily unable to process the request. Try your call again.

HTTP Status Code: 500

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

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

DetectText

Detects text in the input image and converts it into machine-readable text.

Pass the input image as base64-encoded image bytes or as a reference to an image in an Amazon S3 bucket. If you use the AWS CLI to call Amazon Rekognition operations, you must pass it as a reference to an image in an Amazon S3 bucket. For the AWS CLI, passing image bytes is not supported. The image must be either a .png or .jpeg formatted file.

The `DetectText` operation returns text in an array of [TextDetection \(p. 394\)](#) elements, `TextDetections`. Each `TextDetection` element provides information about a single word or line of text that was detected in the image.

A word is one or more ISO basic latin script characters that are not separated by spaces. `DetectText` can detect up to 50 words in an image.

A line is a string of equally spaced words. A line isn't necessarily a complete sentence. For example, a driver's license number is detected as a line. A line ends when there is no aligned text after it. Also, a line ends when there is a large gap between words, relative to the length of the words. This means, depending on the gap between words, Amazon Rekognition may detect multiple lines in text aligned in the same direction. Periods don't represent the end of a line. If a sentence spans multiple lines, the `DetectText` operation returns multiple lines.

To determine whether a `TextDetection` element is a line of text or a word, use the `TextDetection` object `Type` field.

To be detected, text must be within +/- 90 degrees orientation of the horizontal axis.

For more information, see [텍스트 감지 \(p. 196\)](#).

Request Syntax

```
{  
    "Image": {  
        "Bytes": blob,  
        "S3Object": {  
            "Bucket": "string",  
            "Name": "string",  
            "Version": "string"  
        }  
    }  
}
```

Request Parameters

The request accepts the following data in JSON format.

[Image \(p. 254\)](#)

The input image as base64-encoded bytes or an Amazon S3 object. If you use the AWS CLI to call Amazon Rekognition operations, you can't pass image bytes.

Type: [Image \(p. 368\)](#) object

Required: Yes

Response Syntax

```
{
```

```
"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"  
    }  
]
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

[TextDetections \(p. 254\)](#)

An array of text that was detected in the input image.

Type: Array of [TextDetection \(p. 394\)](#) objects

Errors

AccessDeniedException

You are not authorized to perform the action.

HTTP Status Code: 400

ImageTooLargeException

The input image size exceeds the allowed limit. For more information, see [Amazon Rekognition의 제한 \(p. 400\)](#).

HTTP Status Code: 400

InternalServerError

Amazon Rekognition experienced a service issue. Try your call again.

HTTP Status Code: 500

InvalidImageFormatException

The provided image format is not supported.

HTTP Status Code: 400

InvalidArgumentException

Input parameter violated a constraint. Validate your parameter before calling the API operation again.

HTTP Status Code: 400

InvalidS3ObjectException

Amazon Rekognition is unable to access the S3 object specified in the request.

HTTP Status Code: 400

ProvisionedThroughputExceededException

The number of requests exceeded your throughput limit. If you want to increase this limit, contact Amazon Rekognition.

HTTP Status Code: 400

ThrottlingException

Amazon Rekognition is temporarily unable to process the request. Try your call again.

HTTP Status Code: 500

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

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

GetCelebrityInfo

Gets the name and additional information about a celebrity based on his or her Amazon Rekognition ID. The additional information is returned as an array of URLs. If there is no additional information about the celebrity, this list is empty.

For more information, see [유명 인사에 대한 정보 얻기 \(p. 183\)](#).

This operation requires permissions to perform the `rekognition:GetCelebrityInfo` action.

Request Syntax

```
{  
    "Id": "string"  
}
```

Request Parameters

The request accepts the following data in JSON format.

[Id \(p. 257\)](#)

The ID for the celebrity. You get the celebrity ID from a call to the [RecognizeCelebrities \(p. 304\)](#) operation, which recognizes celebrities in an image.

Type: String

Pattern: [0-9A-Za-z]*

Required: Yes

Response Syntax

```
{  
    "Name": "string",  
    "Urls": [ "string" ]  
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

[Name \(p. 257\)](#)

The name of the celebrity.

Type: String

[Urls \(p. 257\)](#)

An array of URLs pointing to additional celebrity information.

Type: Array of strings

Errors

AccessDeniedException

You are not authorized to perform the action.

HTTP Status Code: 400

InternalServerError

Amazon Rekognition experienced a service issue. Try your call again.

HTTP Status Code: 500

InvalidParameterException

Input parameter violated a constraint. Validate your parameter before calling the API operation again.

HTTP Status Code: 400

ProvisionedThroughputExceededException

The number of requests exceeded your throughput limit. If you want to increase this limit, contact Amazon Rekognition.

HTTP Status Code: 400

ResourceNotFoundException

The collection specified in the request cannot be found.

HTTP Status Code: 400

ThrottlingException

Amazon Rekognition is temporarily unable to process the request. Try your call again.

HTTP Status Code: 500

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

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

GetCelebrityRecognition

Gets the celebrity recognition results for a Amazon Rekognition Video analysis started by [StartCelebrityRecognition \(p. 315\)](#).

Celebrity recognition in a video is an asynchronous operation. Analysis is started by a call to [StartCelebrityRecognition \(p. 315\)](#) which returns a job identifier (`JobId`). When the celebrity recognition operation finishes, Amazon Rekognition Video publishes a completion status to the Amazon Simple Notification Service topic registered in the initial call to `StartCelebrityRecognition`. To get the results of the celebrity recognition analysis, first check that the status value published to the Amazon SNS topic is `SUCCEEDED`. If so, call `GetCelebrityDetection` and pass the job identifier (`JobId`) from the initial call to `StartCelebrityDetection`.

For more information, see [저장된 비디오 작업 \(p. 57\)](#).

`GetCelebrityRecognition` returns detected celebrities and the time(s) they are detected in an array (`Celebrities`) of [CelebrityRecognition \(p. 349\)](#) objects. Each `CelebrityRecognition` contains information about the celebrity in a [CelebrityDetail \(p. 347\)](#) object and the time, `Timestamp`, the celebrity was detected.

Note

`GetCelebrityRecognition` only returns the default facial attributes (`BoundingBox`, `Confidence`, `Landmarks`, `Pose`, and `Quality`). The other facial attributes listed in the `Face` object of the following response syntax are not returned. For more information, see [FaceDetail \(p. 359\)](#).

By default, the `Celebrities` array is sorted by time (milliseconds from the start of the video). You can also sort the array by celebrity by specifying the value `ID` in the `SortBy` input parameter.

The `CelebrityDetail` object includes the celebrity identifier and additional information urls. If you don't store the additional information urls, you can get them later by calling [GetCelebrityInfo \(p. 257\)](#) with the celebrity identifier.

No information is returned for faces not recognized as celebrities.

Use `MaxResults` parameter to limit the number of labels returned. If there are more results than specified in `MaxResults`, the value of `NextToken` in the operation response contains a pagination token for getting the next set of results. To get the next page of results, call `GetCelebrityDetection` and populate the `NextToken` request parameter with the token value returned from the previous call to `GetCelebrityRecognition`.

Request Syntax

```
{  
    "JobId": "string",  
    "MaxResults": number,  
    "NextToken": "string",  
    "SortBy": "string"  
}
```

Request Parameters

The request accepts the following data in JSON format.

JobId (p. 259)

Job identifier for the required celebrity recognition analysis. You can get the job identifier from a call to `StartCelebrityRecognition`.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 64.

Pattern: ^[a-zA-Z0-9-_]+\$

Required: Yes

[MaxResults \(p. 259\)](#)

Maximum number of results to return per paginated call. The largest value you can specify is 1000. If you specify a value greater than 1000, a maximum of 1000 results is returned. The default value is 1000.

Type: Integer

Valid Range: Minimum value of 1.

Required: No

[NextToken \(p. 259\)](#)

If the previous response was incomplete (because there is more recognized celebrities to retrieve), Amazon Rekognition Video returns a pagination token in the response. You can use this pagination token to retrieve the next set of celebrities.

Type: String

Length Constraints: Maximum length of 255.

Required: No

[SortBy \(p. 259\)](#)

Sort to use for celebrities returned in `Celebrities` field. Specify `ID` to sort by the celebrity identifier, specify `TIMESTAMP` to sort by the time the celebrity was recognized.

Type: String

Valid Values: `ID` | `TIMESTAMP`

Required: No

Response Syntax

```
{  
    "Celebrities": [  
        {  
            "Celebrity": {  
                "BoundingBox": {  
                    "Height": number,  
                    "Left": number,  
                    "Top": number,  
                    "Width": number  
                },  
                "Confidence": number,  
                "Face": {  
                    "AgeRange": {  
                        "High": number,  
                        "Low": number  
                    },  
                    "Beard": {  
                        "Confidence": number,  
                        "Type": string  
                    }  
                }  
            }  
        }  
    ]  
}
```

```
        "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,
    "Name": string,
    "Urls": [ string ]
},
"Timestamp": number
}
```

```
],
"JobStatus": "string",
"NextToken": "string",
"StatusMessage": "string",
"VideoMetadata": {
  "Codec": "string",
  "DurationMillis": number,
  "Format": "string",
  "FrameHeight": number,
  "FrameRate": number,
  "FrameWidth": number
}
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

[Celebrities \(p. 260\)](#)

Array of celebrities recognized in the video.

Type: Array of [CelebrityRecognition \(p. 349\)](#) objects

[JobStatus \(p. 260\)](#)

The current status of the celebrity recognition job.

Type: String

Valid Values: IN_PROGRESS | SUCCEEDED | FAILED

[NextToken \(p. 260\)](#)

If the response is truncated, Amazon Rekognition Video returns this token that you can use in the subsequent request to retrieve the next set of celebrities.

Type: String

Length Constraints: Maximum length of 255.

[StatusMessage \(p. 260\)](#)

If the job fails, StatusMessage provides a descriptive error message.

Type: String

[VideoMetadata \(p. 260\)](#)

Information about a video that Amazon Rekognition Video analyzed. Videometadata is returned in every page of paginated responses from a Amazon Rekognition Video operation.

Type: [VideoMetadata \(p. 398\)](#) object

Errors

AccessDeniedException

You are not authorized to perform the action.

HTTP Status Code: 400

InternalServerError

Amazon Rekognition experienced a service issue. Try your call again.

HTTP Status Code: 500

InvalidPaginationTokenException

Pagination token in the request is not valid.

HTTP Status Code: 400

InvalidParameterException

Input parameter violated a constraint. Validate your parameter before calling the API operation again.

HTTP Status Code: 400

ProvisionedThroughputExceededException

The number of requests exceeded your throughput limit. If you want to increase this limit, contact Amazon Rekognition.

HTTP Status Code: 400

ResourceNotFoundException

The collection specified in the request cannot be found.

HTTP Status Code: 400

ThrottlingException

Amazon Rekognition is temporarily unable to process the request. Try your call again.

HTTP Status Code: 500

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

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

GetContentModeration

Gets the content moderation analysis results for a Amazon Rekognition Video analysis started by [StartContentModeration](#) (p. 318).

Content moderation analysis of a video is an asynchronous operation. You start analysis by calling [StartContentModeration](#) (p. 318), which returns a job identifier (`JobId`). When analysis finishes, Amazon Rekognition Video publishes a completion status to the Amazon Simple Notification Service topic registered in the initial call to `StartContentModeration`. To get the results of the content moderation analysis, first check that the status value published to the Amazon SNS topic is `SUCCEEDED`. If so, call `GetCelebrityDetection` and pass the job identifier (`JobId`) from the initial call to `StartCelebrityDetection`.

For more information, see [저장된 비디오 작업](#) (p. 57).

`GetContentModeration` returns detected content moderation labels, and the time they are detected, in an array, `ModerationLabels`, of [ContentModerationDetection](#) (p. 353) objects.

By default, the moderated labels are returned sorted by time, in milliseconds from the start of the video. You can also sort them by moderated label by specifying `NAME` for the `SortBy` input parameter.

Since video analysis can return a large number of results, use the `MaxResults` parameter to limit the number of labels returned in a single call to `GetContentModeration`. If there are more results than specified in `MaxResults`, the value of `NextToken` in the operation response contains a pagination token for getting the next set of results. To get the next page of results, call `GetContentModeration` and populate the `NextToken` request parameter with the value of `NextToken` returned from the previous call to `GetContentModeration`.

For more information, see [비안전 콘텐츠 감지](#) (p. 187).

Request Syntax

```
{  
    "JobId": "string",  
    "MaxResults": number,  
    "NextToken": "string",  
    "SortBy": "string"  
}
```

Request Parameters

The request accepts the following data in JSON format.

`JobId` (p. 264)

The identifier for the content moderation job. Use `JobId` to identify the job in a subsequent call to `GetContentModeration`.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 64.

Pattern: ^[a-zA-Z0-9-_]+\$

Required: Yes

`MaxResults` (p. 264)

Maximum number of results to return per paginated call. The largest value you can specify is 1000. If you specify a value greater than 1000, a maximum of 1000 results is returned. The default value is 1000.

Type: Integer

Valid Range: Minimum value of 1.

Required: No

[NextToken \(p. 264\)](#)

If the previous response was incomplete (because there is more data to retrieve), Amazon Rekognition returns a pagination token in the response. You can use this pagination token to retrieve the next set of content moderation labels.

Type: String

Length Constraints: Maximum length of 255.

Required: No

[SortBy \(p. 264\)](#)

Sort to use for elements in the `ModerationLabelDetections` array. Use `TIMESTAMP` to sort array elements by the time labels are detected. Use `NAME` to alphabetically group elements for a label together. Within each label group, the array element are sorted by detection confidence. The default sort is by `TIMESTAMP`.

Type: String

Valid Values: `NAME` | `TIMESTAMP`

Required: No

Response Syntax

```
{  
    "JobStatus": "string",  
    "ModerationLabels": [  
        {  
            "ModerationLabel": {  
                "Confidence": number,  
                "Name": "string",  
                "ParentName": "string"  
            },  
            "Timestamp": number  
        }  
    ],  
    "NextToken": "string",  
    "StatusMessage": "string",  
    "VideoMetadata": {  
        "Codec": "string",  
        "DurationMillis": number,  
        "Format": "string",  
        "FrameHeight": number,  
        "FrameRate": number,  
        "FrameWidth": number  
    }  
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

[JobStatus \(p. 265\)](#)

The current status of the content moderation job.

Type: String

Valid Values: IN_PROGRESS | SUCCEEDED | FAILED

[ModerationLabels \(p. 265\)](#)

The detected moderation labels and the time(s) they were detected.

Type: Array of [ContentModerationDetection \(p. 353\)](#) objects

[NextToken \(p. 265\)](#)

If the response is truncated, Amazon Rekognition Video returns this token that you can use in the subsequent request to retrieve the next set of moderation labels.

Type: String

Length Constraints: Maximum length of 255.

[StatusMessage \(p. 265\)](#)

If the job fails, StatusMessage provides a descriptive error message.

Type: String

[VideoMetadata \(p. 265\)](#)

Information about a video that Amazon Rekognition analyzed. Videometadata is returned in every page of paginated responses from GetContentModeration.

Type: [VideoMetadata \(p. 398\)](#) object

Errors

AccessDeniedException

You are not authorized to perform the action.

HTTP Status Code: 400

InternalServerError

Amazon Rekognition experienced a service issue. Try your call again.

HTTP Status Code: 500

InvalidPaginationTokenException

Pagination token in the request is not valid.

HTTP Status Code: 400

InvalidParameterException

Input parameter violated a constraint. Validate your parameter before calling the API operation again.

HTTP Status Code: 400

ProvisionedThroughputExceededException

The number of requests exceeded your throughput limit. If you want to increase this limit, contact Amazon Rekognition.

HTTP Status Code: 400
ResourceNotFoundException

The collection specified in the request cannot be found.

HTTP Status Code: 400
ThrottlingException

Amazon Rekognition is temporarily unable to process the request. Try your call again.

HTTP Status Code: 500

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

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

GetFaceDetection

Gets face detection results for a Amazon Rekognition Video analysis started by [StartFaceDetection \(p. 322\)](#).

Face detection with Amazon Rekognition Video is an asynchronous operation. You start face detection by calling [StartFaceDetection \(p. 322\)](#) which returns a job identifier (`JobId`). When the face detection operation finishes, Amazon Rekognition Video publishes a completion status to the Amazon Simple Notification Service topic registered in the initial call to `StartFaceDetection`. To get the results of the face detection operation, first check that the status value published to the Amazon SNS topic is `SUCCEEDED`. If so, call [GetFaceDetection \(p. 268\)](#) and pass the job identifier (`JobId`) from the initial call to `StartFaceDetection`.

`GetFaceDetection` returns an array of detected faces (`Faces`) sorted by the time the faces were detected.

Use `MaxResults` parameter to limit the number of labels returned. If there are more results than specified in `MaxResults`, the value of `NextToken` in the operation response contains a pagination token for getting the next set of results. To get the next page of results, call `GetFaceDetection` and populate the `NextToken` request parameter with the token value returned from the previous call to `GetFaceDetection`.

Request Syntax

```
{  
    "JobId": "string",  
    "MaxResults": number,  
    "NextToken": "string"  
}
```

Request Parameters

The request accepts the following data in JSON format.

[JobId \(p. 268\)](#)

Unique identifier for the face detection job. The `JobId` is returned from `StartFaceDetection`.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 64.

Pattern: ^[a-zA-Z0-9-_]+\$

Required: Yes

[MaxResults \(p. 268\)](#)

Maximum number of results to return per paginated call. The largest value you can specify is 1000. If you specify a value greater than 1000, a maximum of 1000 results is returned. The default value is 1000.

Type: Integer

Valid Range: Minimum value of 1.

Required: No

NextToken (p. 268)

If the previous response was incomplete (because there are more faces to retrieve), Amazon Rekognition Video returns a pagination token in the response. You can use this pagination token to retrieve the next set of faces.

Type: String

Length Constraints: Maximum length of 255.

Required: No

Response Syntax

```
{  
    "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",
    "DurationMillis": number,
    "Format": "string",
    "FrameHeight": number,
    "FrameRate": number,
    "FrameWidth": number
}
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

[Faces \(p. 269\)](#)

An array of faces detected in the video. Each element contains a detected face's details and the time, in milliseconds from the start of the video, the face was detected.

Type: Array of [FaceDetection \(p. 362\)](#) objects

[JobStatus \(p. 269\)](#)

The current status of the face detection job.

Type: String

Valid Values: IN_PROGRESS | SUCCEEDED | FAILED

[NextToken \(p. 269\)](#)

If the response is truncated, Amazon Rekognition returns this token that you can use in the subsequent request to retrieve the next set of faces.

Type: String

Length Constraints: Maximum length of 255.

[StatusMessage \(p. 269\)](#)

If the job fails, `StatusMessage` provides a descriptive error message.

Type: String

[VideoMetadata \(p. 269\)](#)

Information about a video that Amazon Rekognition Video analyzed. `Videodata` is returned in every page of paginated responses from a Amazon Rekognition video operation.

Type: [VideoMetadata \(p. 398\)](#) object

Errors

`AccessDeniedException`

You are not authorized to perform the action.

HTTP Status Code: 400

`InternalServerError`

Amazon Rekognition experienced a service issue. Try your call again.

HTTP Status Code: 500

`InvalidPaginationTokenException`

Pagination token in the request is not valid.

HTTP Status Code: 400

`InvalidParameterException`

Input parameter violated a constraint. Validate your parameter before calling the API operation again.

HTTP Status Code: 400

`ProvisionedThroughputExceededException`

The number of requests exceeded your throughput limit. If you want to increase this limit, contact Amazon Rekognition.

HTTP Status Code: 400

`ResourceNotFoundException`

The collection specified in the request cannot be found.

HTTP Status Code: 400

`ThrottlingException`

Amazon Rekognition is temporarily unable to process the request. Try your call again.

HTTP Status Code: 500

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)

- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V2

GetFaceSearch

Gets the face search results for Amazon Rekognition Video face search started by [StartFaceSearch \(p. 326\)](#). The search returns faces in a collection that match the faces of persons detected in a video. It also includes the time(s) that faces are matched in the video.

Face search in a video is an asynchronous operation. You start face search by calling to [StartFaceSearch \(p. 326\)](#) which returns a job identifier (`JobId`). When the search operation finishes, Amazon Rekognition Video publishes a completion status to the Amazon Simple Notification Service topic registered in the initial call to `StartFaceSearch`. To get the search results, first check that the `status` value published to the Amazon SNS topic is `SUCCEEDED`. If so, call `GetFaceSearch` and pass the job identifier (`JobId`) from the initial call to `StartFaceSearch`.

For more information, see [모음에서 얼굴 검색 \(p. 127\)](#).

The search results are returned in an array, `Persons`, of [PersonMatch \(p. 384\)](#) objects. Each `PersonMatch` element contains details about the matching faces in the input collection, person information (facial attributes, bounding boxes, and person identifier) for the matched person, and the time the person was matched in the video.

Note

`GetFaceSearch` only returns the default facial attributes (`BoundingBox`, `Confidence`, `Landmarks`, `Pose`, and `Quality`). The other facial attributes listed in the `Face` object of the following response syntax are not returned. For more information, see [FaceDetail \(p. 359\)](#).

By default, the `Persons` array is sorted by the time, in milliseconds from the start of the video, persons are matched. You can also sort by persons by specifying `INDEX` for the `SORTBY` input parameter.

Request Syntax

```
{  
    "JobId": "string",  
    "MaxResults": number,  
    "NextToken": "string",  
    "SortBy": "string"  
}
```

Request Parameters

The request accepts the following data in JSON format.

`JobId` (p. 273)

The job identifier for the search request. You get the job identifier from an initial call to `StartFaceSearch`.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 64.

Pattern: ^[a-zA-Z0-9-_]+\$

Required: Yes

`MaxResults` (p. 273)

Maximum number of results to return per paginated call. The largest value you can specify is 1000. If you specify a value greater than 1000, a maximum of 1000 results is returned. The default value is 1000.

Type: Integer

Valid Range: Minimum value of 1.

Required: No

[NextToken \(p. 273\)](#)

If the previous response was incomplete (because there is more search results to retrieve), Amazon Rekognition Video returns a pagination token in the response. You can use this pagination token to retrieve the next set of search results.

Type: String

Length Constraints: Maximum length of 255.

Required: No

[SortBy \(p. 273\)](#)

Sort to use for grouping faces in the response. Use `TIMESTAMP` to group faces by the time that they are recognized. Use `INDEX` to sort by recognized faces.

Type: String

Valid Values: `INDEX` | `TIMESTAMP`

Required: No

Response Syntax

```
{  
    "JobStatus": "string",  
    "NextToken": "string",  
    "Persons": [  
        {  
            "FaceMatches": [  
                {  
                    "Face": {  
                        "BoundingBox": {  
                            "Height": number,  
                            "Left": number,  
                            "Top": number,  
                            "Width": number  
                        },  
                        "Confidence": number,  
                        "ExternalImageId": "string",  
                        "FaceId": "string",  
                        "ImageId": "string"  
                    },  
                    "Similarity": number  
                }  
            ],  
            "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
},
"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",
        "DurationMillis": number,
        "Format": "string",
        "FrameHeight": number,
        "FrameRate": number,
        "FrameWidth": number
    }
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

[JobStatus \(p. 274\)](#)

The current status of the face search job.

Type: String

Valid Values: IN_PROGRESS | SUCCEEDED | FAILED

[NextToken \(p. 274\)](#)

If the response is truncated, Amazon Rekognition Video returns this token that you can use in the subsequent request to retrieve the next set of search results.

Type: String

Length Constraints: Maximum length of 255.

[Persons \(p. 274\)](#)

An array of persons, [PersonMatch \(p. 384\)](#), in the video whose face(s) match the face(s) in an Amazon Rekognition collection. It also includes time information for when persons are matched in the video. You specify the input collection in an initial call to [StartFaceSearch](#). Each Persons element includes a time the person was matched, face match details ([FaceMatches](#)) for matching faces in the collection, and person information ([Person](#)) for the matched person.

Type: Array of [PersonMatch \(p. 384\)](#) objects

[StatusMessage \(p. 274\)](#)

If the job fails, StatusMessage provides a descriptive error message.

Type: String

[VideoMetadata \(p. 274\)](#)

Information about a video that Amazon Rekognition analyzed. VideoMetadata is returned in every page of paginated responses from a Amazon Rekognition Video operation.

Type: [VideoMetadata \(p. 398\)](#) object

Errors

AccessDeniedException

You are not authorized to perform the action.

HTTP Status Code: 400
InternalServerError

Amazon Rekognition experienced a service issue. Try your call again.

HTTP Status Code: 500
InvalidPaginationTokenException

Pagination token in the request is not valid.

HTTP Status Code: 400
InvalidParameterException

Input parameter violated a constraint. Validate your parameter before calling the API operation again.

HTTP Status Code: 400
ProvisionedThroughputExceededException

The number of requests exceeded your throughput limit. If you want to increase this limit, contact Amazon Rekognition.

HTTP Status Code: 400
ResourceNotFoundException

The collection specified in the request cannot be found.

HTTP Status Code: 400
ThrottlingException

Amazon Rekognition is temporarily unable to process the request. Try your call again.

HTTP Status Code: 500

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

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

GetLabelDetection

Gets the label detection results of a Amazon Rekognition Video analysis started by [StartLabelDetection \(p. 330\)](#).

The label detection operation is started by a call to [StartLabelDetection \(p. 330\)](#) which returns a job identifier (`JobId`). When the label detection operation finishes, Amazon Rekognition publishes a completion status to the Amazon Simple Notification Service topic registered in the initial call to `StartLabelDetection`. To get the results of the label detection operation, first check that the status value published to the Amazon SNS topic is `SUCCEEDED`. If so, call [GetLabelDetection \(p. 278\)](#) and pass the job identifier (`JobId`) from the initial call to `StartLabelDetection`.

`GetLabelDetection` returns an array of detected labels (`Labels`) sorted by the time the labels were detected. You can also sort by the label name by specifying `NAME` for the `SortBy` input parameter.

The labels returned include the label name, the percentage confidence in the accuracy of the detected label, and the time the label was detected in the video.

Use `MaxResults` parameter to limit the number of labels returned. If there are more results than specified in `MaxResults`, the value of `NextToken` in the operation response contains a pagination token for getting the next set of results. To get the next page of results, call `GetLabelDetection` and populate the `NextToken` request parameter with the token value returned from the previous call to `GetLabelDetection`.

Note

`GetLabelDetection` doesn't return a hierarchical taxonomy, or bounding box information, for detected labels. `GetLabelDetection` returns `null` for the `Parents` and `Instances` attributes of the [Label \(p. 373\)](#) object which is returned in the `Labels` array.

Request Syntax

```
{  
    "JobId": "string",  
    "MaxResults": number,  
    "NextToken": "string",  
    "SortBy": "string"  
}
```

Request Parameters

The request accepts the following data in JSON format.

`JobId` (p. 278)

Job identifier for the label detection operation for which you want results returned. You get the job identifier from an initial call to `StartLabelDetection`.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 64.

Pattern: `^[a-zA-Z0-9-_]+$`

Required: Yes

`MaxResults` (p. 278)

Maximum number of results to return per paginated call. The largest value you can specify is 1000. If you specify a value greater than 1000, a maximum of 1000 results is returned. The default value is 1000.

Type: Integer

Valid Range: Minimum value of 1.

Required: No

[NextToken \(p. 278\)](#)

If the previous response was incomplete (because there are more labels to retrieve), Amazon Rekognition Video returns a pagination token in the response. You can use this pagination token to retrieve the next set of labels.

Type: String

Length Constraints: Maximum length of 255.

Required: No

[SortBy \(p. 278\)](#)

Sort to use for elements in the `Labels` array. Use `TIMESTAMP` to sort array elements by the time labels are detected. Use `NAME` to alphabetically group elements for a label together. Within each label group, the array elements are sorted by detection confidence. The default sort is by `TIMESTAMP`.

Type: String

Valid Values: `NAME` | `TIMESTAMP`

Required: No

Response Syntax

```
{  
    "JobStatus": "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",  
    }  
}
```

```
    "DurationMillis": number,
    "Format": "string",
    "FrameHeight": number,
    "FrameRate": number,
    "FrameWidth": number
}
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

[JobStatus \(p. 279\)](#)

The current status of the label detection job.

Type: String

Valid Values: IN_PROGRESS | SUCCEEDED | FAILED

[Labels \(p. 279\)](#)

An array of labels detected in the video. Each element contains the detected label and the time, in milliseconds from the start of the video, that the label was detected.

Type: Array of [LabelDetection \(p. 375\)](#) objects

[NextToken \(p. 279\)](#)

If the response is truncated, Amazon Rekognition Video returns this token that you can use in the subsequent request to retrieve the next set of labels.

Type: String

Length Constraints: Maximum length of 255.

[StatusMessage \(p. 279\)](#)

If the job fails, StatusMessage provides a descriptive error message.

Type: String

[VideoMetadata \(p. 279\)](#)

Information about a video that Amazon Rekognition Video analyzed. Videometadata is returned in every page of paginated responses from a Amazon Rekognition video operation.

Type: [VideoMetadata \(p. 398\)](#) object

Errors

AccessDeniedException

You are not authorized to perform the action.

HTTP Status Code: 400

InternalServerError

Amazon Rekognition experienced a service issue. Try your call again.

HTTP Status Code: 500

InvalidPaginationTokenException

Pagination token in the request is not valid.

HTTP Status Code: 400

InvalidParameterException

Input parameter violated a constraint. Validate your parameter before calling the API operation again.

HTTP Status Code: 400

ProvisionedThroughputExceededException

The number of requests exceeded your throughput limit. If you want to increase this limit, contact Amazon Rekognition.

HTTP Status Code: 400

ResourceNotFoundException

The collection specified in the request cannot be found.

HTTP Status Code: 400

ThrottlingException

Amazon Rekognition is temporarily unable to process the request. Try your call again.

HTTP Status Code: 500

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

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

GetPersonTracking

Gets the path tracking results of a Amazon Rekognition Video analysis started by [StartPersonTracking \(p. 334\)](#).

The person path tracking operation is started by a call to `StartPersonTracking` which returns a job identifier (`JobId`). When the operation finishes, Amazon Rekognition Video publishes a completion status to the Amazon Simple Notification Service topic registered in the initial call to `StartPersonTracking`.

To get the results of the person path tracking operation, first check that the status value published to the Amazon SNS topic is `SUCCEEDED`. If so, call [GetPersonTracking \(p. 282\)](#) and pass the job identifier (`JobId`) from the initial call to `StartPersonTracking`.

`GetPersonTracking` returns an array, `Persons`, of tracked persons and the time(s) their paths were tracked in the video.

Note

`GetPersonTracking` only returns the default facial attributes (`BoundingBox`, `Confidence`, `Landmarks`, `Pose`, and `Quality`). The other facial attributes listed in the `Face` object of the following response syntax are not returned.

For more information, see [FaceDetail \(p. 359\)](#).

By default, the array is sorted by the time(s) a person's path is tracked in the video. You can sort by tracked persons by specifying `INDEX` for the `SortBy` input parameter.

Use the `MaxResults` parameter to limit the number of items returned. If there are more results than specified in `MaxResults`, the value of `NextToken` in the operation response contains a pagination token for getting the next set of results. To get the next page of results, call `GetPersonTracking` and populate the `NextToken` request parameter with the token value returned from the previous call to `GetPersonTracking`.

Request Syntax

```
{  
    "JobId": "string",  
    "MaxResults": number,  
    "NextToken": "string",  
    "SortBy": "string"  
}
```

Request Parameters

The request accepts the following data in JSON format.

JobId (p. 282)

The identifier for a job that tracks persons in a video. You get the `JobID` from a call to `StartPersonTracking`.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 64.

Pattern: `^[a-zA-Z0-9-_]+$`

Required: Yes

[MaxResults \(p. 282\)](#)

Maximum number of results to return per paginated call. The largest value you can specify is 1000. If you specify a value greater than 1000, a maximum of 1000 results is returned. The default value is 1000.

Type: Integer

Valid Range: Minimum value of 1.

Required: No

[NextToken \(p. 282\)](#)

If the previous response was incomplete (because there are more persons to retrieve), Amazon Rekognition Video returns a pagination token in the response. You can use this pagination token to retrieve the next set of persons.

Type: String

Length Constraints: Maximum length of 255.

Required: No

[SortBy \(p. 282\)](#)

Sort to use for elements in the `Persons` array. Use `TIMESTAMP` to sort array elements by the time persons are detected. Use `INDEX` to sort by the tracked persons. If you sort by `INDEX`, the array elements for each person are sorted by detection confidence. The default sort is by `TIMESTAMP`.

Type: String

Valid Values: `INDEX` | `TIMESTAMP`

Required: No

Response Syntax

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

```
        "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",
        "DurationMillis": number,
        "Format": "string",
        "FrameHeight": number,
        "FrameRate": number,
```

```
    "FrameWidth": number
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

[JobStatus \(p. 283\)](#)

The current status of the person tracking job.

Type: String

Valid Values: IN_PROGRESS | SUCCEEDED | FAILED

[NextToken \(p. 283\)](#)

If the response is truncated, Amazon Rekognition Video returns this token that you can use in the subsequent request to retrieve the next set of persons.

Type: String

Length Constraints: Maximum length of 255.

[Persons \(p. 283\)](#)

An array of the persons detected in the video and the time(s) their path was tracked throughout the video. An array element will exist for each time a person's path is tracked.

Type: Array of [PersonDetection \(p. 383\)](#) objects

[StatusMessage \(p. 283\)](#)

If the job fails, `StatusMessage` provides a descriptive error message.

Type: String

[VideoMetadata \(p. 283\)](#)

Information about a video that Amazon Rekognition Video analyzed. `Videometadata` is returned in every page of paginated responses from a Amazon Rekognition Video operation.

Type: [VideoMetadata \(p. 398\)](#) object

Errors

AccessDeniedException

You are not authorized to perform the action.

HTTP Status Code: 400

InternalServerError

Amazon Rekognition experienced a service issue. Try your call again.

HTTP Status Code: 500

InvalidPaginationTokenException

Pagination token in the request is not valid.

HTTP Status Code: 400

InvalidParameterException

Input parameter violated a constraint. Validate your parameter before calling the API operation again.

HTTP Status Code: 400

ProvisionedThroughputExceededException

The number of requests exceeded your throughput limit. If you want to increase this limit, contact Amazon Rekognition.

HTTP Status Code: 400

ResourceNotFoundException

The collection specified in the request cannot be found.

HTTP Status Code: 400

ThrottlingException

Amazon Rekognition is temporarily unable to process the request. Try your call again.

HTTP Status Code: 500

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

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

IndexFaces

Detects faces in the input image and adds them to the specified collection.

Amazon Rekognition doesn't save the actual faces that are detected. Instead, the underlying detection algorithm first detects the faces in the input image. For each face, the algorithm extracts facial features into a feature vector, and stores it in the backend database. Amazon Rekognition uses feature vectors when it performs face match and search operations using the [SearchFaces \(p. 308\)](#) and [SearchFacesByImage \(p. 311\)](#) operations.

For more information, see [모음에 얼굴 추가 \(p. 137\)](#).

To get the number of faces in a collection, call [DescribeCollection \(p. 236\)](#).

If you're using version 1.0 of the face detection model, `IndexFaces` indexes the 15 largest faces in the input image. Later versions of the face detection model index the 100 largest faces in the input image.

If you're using version 4 or later of the face model, image orientation information is not returned in the `OrientationCorrection` field.

To determine which version of the model you're using, call [DescribeCollection \(p. 236\)](#) and supply the collection ID. You can also get the model version from the value of `FaceModelVersion` in the response from `IndexFaces`

For more information, see [모델 버전 관리 \(p. 9\)](#).

If you provide the optional `ExternalImageID` for the input image you provided, Amazon Rekognition associates this ID with all faces that it detects. When you call the [ListFaces \(p. 298\)](#) operation, the response returns the external ID. You can use this external image ID to create a client-side index to associate the faces with each image. You can then use the index to find all faces in an image.

You can specify the maximum number of faces to index with the `MaxFaces` input parameter. This is useful when you want to index the largest faces in an image and don't want to index smaller faces, such as those belonging to people standing in the background.

The `QualityFilter` input parameter allows you to filter out detected faces that don't meet the required quality bar chosen by Amazon Rekognition. The quality bar is based on a variety of common use cases. By default, `IndexFaces` filters detected faces. You can also explicitly filter detected faces by specifying `AUTO` for the value of `QualityFilter`. If you do not want to filter detected faces, specify `NONE`.

Note

To use quality filtering, you need a collection associated with version 3 of the face model. To get the version of the face model associated with a collection, call [DescribeCollection \(p. 236\)](#).

Information about faces detected in an image, but not indexed, is returned in an array of [UnindexedFace \(p. 396\)](#) objects, `UnindexedFaces`. Faces aren't indexed for reasons such as:

- The number of faces detected exceeds the value of the `MaxFaces` request parameter.
- The face is too small compared to the image dimensions.
- The face is too blurry.
- The image is too dark.
- The face has an extreme pose.

In response, the `IndexFaces` operation returns an array of metadata for all detected faces, `FaceRecords`. This includes:

- The bounding box, `BoundingBox`, of the detected face.

- A confidence value, `Confidence`, which indicates the confidence that the bounding box contains a face.
- A face ID, `faceId`, assigned by the service for each face that's detected and stored.
- An image ID, `ImageId`, assigned by the service for the input image.

If you request all facial attributes (by using the `detectionAttributes` parameter), Amazon Rekognition returns detailed facial attributes, such as facial landmarks (for example, location of eye and mouth) and other facial attributes like gender. If you provide the same image, specify the same collection, and use the same external ID in the `IndexFaces` operation, Amazon Rekognition doesn't save duplicate face metadata.

The input image is passed either as base64-encoded image bytes, or as a reference to an image in an Amazon S3 bucket. If you use the AWS CLI to call Amazon Rekognition operations, passing image bytes isn't supported. The image must be formatted as a PNG or JPEG file.

This operation requires permissions to perform the `rekognition:IndexFaces` action.

Request Syntax

```
{  
    "CollectionId": "string",  
    "DetectionAttributes": [ "string" ],  
    "ExternalImageId": "string",  
    "Image": {  
        "Bytes": blob,  
        "S3Object": {  
            "Bucket": "string",  
            "Name": "string",  
            "Version": "string"  
        }  
    },  
    "MaxFaces": number,  
    "QualityFilter": "string"  
}
```

Request Parameters

The request accepts the following data in JSON format.

CollectionId ([p. 288](#))

The ID of an existing collection to which you want to add the faces that are detected in the input images.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 255.

Pattern: [a-zA-Z0-9_.\-\-]+

Required: Yes

DetectionAttributes ([p. 288](#))

An array of facial attributes that you want to be returned. This can be the default list of attributes or all attributes. If you don't specify a value for `Attributes` or if you specify `["DEFAULT"]`, the API returns the following subset of facial attributes: `BoundingBox`, `Confidence`, `Pose`, `Quality`, and `Landmarks`. If you provide `["ALL"]`, all facial attributes are returned, but the operation takes longer to complete.

If you provide both, ["ALL" , "DEFAULT"], the service uses a logical AND operator to determine which attributes to return (in this case, all attributes).

Type: Array of strings

Valid Values: DEFAULT | ALL

Required: No

[ExternalImageId \(p. 288\)](#)

The ID you want to assign to all the faces detected in the image.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 255.

Pattern: [a-zA-Z0-9_.\-\:]⁺

Required: No

[Image \(p. 288\)](#)

The input image as base64-encoded bytes or an S3 object. If you use the AWS CLI to call Amazon Rekognition operations, passing base64-encoded image bytes isn't supported.

Type: [Image \(p. 368\)](#) object

Required: Yes

[MaxFaces \(p. 288\)](#)

The maximum number of faces to index. The value of `MaxFaces` must be greater than or equal to 1. `IndexFaces` returns no more than 100 detected faces in an image, even if you specify a larger value for `MaxFaces`.

If `IndexFaces` detects more faces than the value of `MaxFaces`, the faces with the lowest quality are filtered out first. If there are still more faces than the value of `MaxFaces`, the faces with the smallest bounding boxes are filtered out (up to the number that's needed to satisfy the value of `MaxFaces`). Information about the unindexed faces is available in the `UnindexedFaces` array.

The faces that are returned by `IndexFaces` are sorted by the largest face bounding box size to the smallest size, in descending order.

`MaxFaces` can be used with a collection associated with any version of the face model.

Type: Integer

Valid Range: Minimum value of 1.

Required: No

[QualityFilter \(p. 288\)](#)

A filter that specifies how much filtering is done to identify faces that are detected with low quality. Filtered faces aren't indexed. If you specify AUTO, filtering prioritizes the identification of faces that don't meet the required quality bar chosen by Amazon Rekognition. The quality bar is based on a variety of common use cases. Low-quality detections can occur for a number of reasons. Some examples are an object that's misidentified as a face, a face that's too blurry, or a face with a pose that's too extreme to use. If you specify NONE, no filtering is performed. The default value is AUTO.

To use quality filtering, the collection you are using must be associated with version 3 of the face model.

Type: String

Valid Values: NONE | AUTO

Required: No

Response Syntax

```
{
    "FaceModelVersion": "string",
    "FaceRecords": [
        {
            "Face": {
                "BoundingBox": {
                    "Height": number,
                    "Left": number,
                    "Top": number,
                    "Width": number
                },
                "Confidence": number,
                "ExternalImageId": "string",
                "FaceId": "string",
                "ImageId": "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,
                    "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" ]
}
]
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

[FaceModelVersion \(p. 290\)](#)

The version number of the face detection model that's associated with the input collection (`CollectionId`).

Type: String

[FaceRecords \(p. 290\)](#)

An array of faces detected and added to the collection. For more information, see [모음에서 얼굴 관리 \(p. 127\)](#).

Type: Array of [FaceRecord \(p. 364\)](#) objects

[OrientationCorrection \(p. 290\)](#)

If your collection is associated with a face detection model that's later than version 3.0, the value of `OrientationCorrection` is always null and no orientation information is returned.

If your collection is associated with a face detection model that's version 3.0 or earlier, the following applies:

- If the input image is in .jpeg format, it might contain exchangeable image file format (Exif) metadata that includes the image's orientation. Amazon Rekognition uses this orientation information to perform image correction - the bounding box coordinates are translated to represent object locations after the orientation information in the Exif metadata is used to correct the image orientation. Images in .png format don't contain Exif metadata. The value of `OrientationCorrection` is null.
- If the image doesn't contain orientation information in its Exif metadata, Amazon Rekognition returns an estimated orientation (ROTATE_0, ROTATE_90, ROTATE_180, ROTATE_270). Amazon Rekognition doesn't perform image correction for images. The bounding box coordinates aren't translated and represent the object locations before the image is rotated.

Bounding box information is returned in the `FaceRecords` array. You can get the version of the face detection model by calling [DescribeCollection \(p. 236\)](#).

Type: String

Valid Values: `ROTATE_0` | `ROTATE_90` | `ROTATE_180` | `ROTATE_270`

[UnindexedFaces \(p. 290\)](#)

An array of faces that were detected in the image but weren't indexed. They weren't indexed because the quality filter identified them as low quality, or the `MaxFaces` request parameter filtered them out. To use the quality filter, you specify the `QualityFilter` request parameter.

Type: Array of [UnindexedFace \(p. 396\)](#) objects

Errors

`AccessDeniedException`

You are not authorized to perform the action.

HTTP Status Code: 400

`ImageTooLargeException`

The input image size exceeds the allowed limit. For more information, see [Amazon Rekognition의 제한 \(p. 400\)](#).

HTTP Status Code: 400

`InternalServerError`

Amazon Rekognition experienced a service issue. Try your call again.

HTTP Status Code: 500

`InvalidImageFormatException`

The provided image format is not supported.

HTTP Status Code: 400

`InvalidParameterException`

Input parameter violated a constraint. Validate your parameter before calling the API operation again.

HTTP Status Code: 400

`InvalidS3ObjectException`

Amazon Rekognition is unable to access the S3 object specified in the request.

HTTP Status Code: 400

ProvisionedThroughputExceededException

The number of requests exceeded your throughput limit. If you want to increase this limit, contact Amazon Rekognition.

HTTP Status Code: 400

ResourceNotFoundException

The collection specified in the request cannot be found.

HTTP Status Code: 400

ThrottlingException

Amazon Rekognition is temporarily unable to process the request. Try your call again.

HTTP Status Code: 500

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

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

ListCollections

Returns list of collection IDs in your account. If the result is truncated, the response also provides a `NextToken` that you can use in the subsequent request to fetch the next set of collection IDs.

For an example, see [모음 나열 \(p. 131\)](#).

This operation requires permissions to perform the `rekognition:ListCollections` action.

Request Syntax

```
{  
    "MaxResults": number,  
    "NextToken": "string"  
}
```

Request Parameters

The request accepts the following data in JSON format.

[MaxResults \(p. 295\)](#)

Maximum number of collection IDs to return.

Type: Integer

Valid Range: Minimum value of 0. Maximum value of 4096.

Required: No

[NextToken \(p. 295\)](#)

Pagination token from the previous response.

Type: String

Length Constraints: Maximum length of 255.

Required: No

Response Syntax

```
{  
    "CollectionIds": [ "string" ],  
    "FaceModelVersions": [ "string" ],  
    "NextToken": "string"  
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

[CollectionIds \(p. 295\)](#)

An array of collection IDs.

Type: Array of strings

Length Constraints: Minimum length of 1. Maximum length of 255.

Pattern: [a-zA-Z0-9_.\-\]+

[FaceModelVersions \(p. 295\)](#)

Version numbers of the face detection models associated with the collections in the array `CollectionIds`. For example, the value of `FaceModelVersions[2]` is the version number for the face detection model used by the collection in `CollectionId[2]`.

Type: Array of strings

[NextToken \(p. 295\)](#)

If the result is truncated, the response provides a `NextToken` that you can use in the subsequent request to fetch the next set of collection IDs.

Type: String

Length Constraints: Maximum length of 255.

Errors

`AccessDeniedException`

You are not authorized to perform the action.

HTTP Status Code: 400

`InternalServerError`

Amazon Rekognition experienced a service issue. Try your call again.

HTTP Status Code: 500

`InvalidPaginationTokenException`

Pagination token in the request is not valid.

HTTP Status Code: 400

`InvalidParameterException`

Input parameter violated a constraint. Validate your parameter before calling the API operation again.

HTTP Status Code: 400

`ProvisionedThroughputExceededException`

The number of requests exceeded your throughput limit. If you want to increase this limit, contact Amazon Rekognition.

HTTP Status Code: 400

`ResourceNotFoundException`

The collection specified in the request cannot be found.

HTTP Status Code: 400

`ThrottlingException`

Amazon Rekognition is temporarily unable to process the request. Try your call again.

HTTP Status Code: 500

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

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

ListFaces

Returns metadata for faces in the specified collection. This metadata includes information such as the bounding box coordinates, the confidence (that the bounding box contains a face), and face ID. For an example, see [모음에 얼굴 나열 \(p. 146\)](#).

This operation requires permissions to perform the `rekognition:ListFaces` action.

Request Syntax

```
{  
    "CollectionId": "string",  
    "MaxResults": number,  
    "NextToken": "string"  
}
```

Request Parameters

The request accepts the following data in JSON format.

[CollectionId \(p. 298\)](#)

ID of the collection from which to list the faces.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 255.

Pattern: [a-zA-Z0-9_.\-\-]+

Required: Yes

[MaxResults \(p. 298\)](#)

Maximum number of faces to return.

Type: Integer

Valid Range: Minimum value of 0. Maximum value of 4096.

Required: No

[NextToken \(p. 298\)](#)

If the previous response was incomplete (because there is more data to retrieve), Amazon Rekognition returns a pagination token in the response. You can use this pagination token to retrieve the next set of faces.

Type: String

Length Constraints: Maximum length of 255.

Required: No

Response Syntax

```
{  
    "FaceModelVersion": "string",  
    "Faces": [  
        ...  
    ]  
}
```

```
{  
    "BoundingBox": {  
        "Height": number,  
        "Left": number,  
        "Top": number,  
        "Width": number  
    },  
    "Confidence": number,  
    "ExternalImageId": "string",  
    "FaceId": "string",  
    "ImageId": "string"  
},  
],  
"NextToken": "string"  
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

[FaceModelVersion \(p. 298\)](#)

Version number of the face detection model associated with the input collection (`CollectionId`).

Type: String

[Faces \(p. 298\)](#)

An array of `Face` objects.

Type: Array of [Face \(p. 357\)](#) objects

[NextToken \(p. 298\)](#)

If the response is truncated, Amazon Rekognition returns this token that you can use in the subsequent request to retrieve the next set of faces.

Type: String

Errors

AccessDeniedException

You are not authorized to perform the action.

HTTP Status Code: 400

InternalServerError

Amazon Rekognition experienced a service issue. Try your call again.

HTTP Status Code: 500

InvalidPaginationTokenException

Pagination token in the request is not valid.

HTTP Status Code: 400

InvalidParameterException

Input parameter violated a constraint. Validate your parameter before calling the API operation again.

HTTP Status Code: 400

ProvisionedThroughputExceededException

The number of requests exceeded your throughput limit. If you want to increase this limit, contact Amazon Rekognition.

HTTP Status Code: 400

ResourceNotFoundException

The collection specified in the request cannot be found.

HTTP Status Code: 400

ThrottlingException

Amazon Rekognition is temporarily unable to process the request. Try your call again.

HTTP Status Code: 500

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

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

ListStreamProcessors

Gets a list of stream processors that you have created with [CreateStreamProcessor \(p. 227\)](#).

Request Syntax

```
{  
    "MaxResults": number,  
    "NextToken": "string"  
}
```

Request Parameters

The request accepts the following data in JSON format.

[MaxResults \(p. 301\)](#)

Maximum number of stream processors you want Amazon Rekognition Video to return in the response. The default is 1000.

Type: Integer

Valid Range: Minimum value of 1.

Required: No

[NextToken \(p. 301\)](#)

If the previous response was incomplete (because there are more stream processors to retrieve), Amazon Rekognition Video returns a pagination token in the response. You can use this pagination token to retrieve the next set of stream processors.

Type: String

Length Constraints: Maximum length of 255.

Required: No

Response Syntax

```
{  
    "NextToken": "string",  
    "StreamProcessors": [  
        {  
            "Name": "string",  
            "Status": "string"  
        }  
    ]  
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

[NextToken \(p. 301\)](#)

If the response is truncated, Amazon Rekognition Video returns this token that you can use in the subsequent request to retrieve the next set of stream processors.

Type: String

Length Constraints: Maximum length of 255.

[StreamProcessors \(p. 301\)](#)

List of stream processors that you have created.

Type: Array of [StreamProcessor \(p. 389\)](#) objects

Errors

AccessDeniedException

You are not authorized to perform the action.

HTTP Status Code: 400

InternalServerError

Amazon Rekognition experienced a service issue. Try your call again.

HTTP Status Code: 500

InvalidPaginationTokenException

Pagination token in the request is not valid.

HTTP Status Code: 400

InvalidParameterException

Input parameter violated a constraint. Validate your parameter before calling the API operation again.

HTTP Status Code: 400

ProvisionedThroughputExceededException

The number of requests exceeded your throughput limit. If you want to increase this limit, contact Amazon Rekognition.

HTTP Status Code: 400

ThrottlingException

Amazon Rekognition is temporarily unable to process the request. Try your call again.

HTTP Status Code: 500

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS Command Line Interface](#)
- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)

- AWS SDK for Java
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V2

RecognizeCelebrities

Returns an array of celebrities recognized in the input image. For more information, see [유명 인사 인식 \(p. 173\)](#).

`RecognizeCelebrities` returns the 100 largest faces in the image. It lists recognized celebrities in the `CelebrityFaces` array and unrecognized faces in the `UnrecognizedFaces` array.

`RecognizeCelebrities` doesn't return celebrities whose faces aren't among the largest 100 faces in the image.

For each celebrity recognized, `RecognizeCelebrities` returns a `Celebrity` object. The `Celebrity` object contains the celebrity name, ID, URL links to additional information, match confidence, and a `ComparedFace` object that you can use to locate the celebrity's face on the image.

Amazon Rekognition doesn't retain information about which images a celebrity has been recognized in. Your application must store this information and use the `Celebrity` ID property as a unique identifier for the celebrity. If you don't store the celebrity name or additional information URLs returned by `RecognizeCelebrities`, you will need the ID to identify the celebrity in a call to the [GetCelebrityInfo \(p. 257\)](#) operation.

You pass the input image either as base64-encoded image bytes or as a reference to an image in an Amazon S3 bucket. If you use the AWS CLI to call Amazon Rekognition operations, passing image bytes is not supported. The image must be either a PNG or JPEG formatted file.

For an example, see [이미지 속 유명 인사 인식 \(p. 173\)](#).

This operation requires permissions to perform the `rekognition:RecognizeCelebrities` operation.

Request Syntax

```
{  
    "Image": {  
        "Bytes": blob,  
        "S3Object": {  
            "Bucket": "string",  
            "Name": "string",  
            "Version": "string"  
        }  
    }  
}
```

Request Parameters

The request accepts the following data in JSON format.

Image (p. 304)

The input image as base64-encoded bytes or an S3 object. If you use the AWS CLI to call Amazon Rekognition operations, passing base64-encoded image bytes is not supported.

Type: [Image \(p. 368\)](#) object

Required: Yes

Response Syntax

```
{  
    "CelebrityFaces": [  
        {  
            "Celebrity": {  
                "Name": "string",  
                "Id": "string",  
                "MatchConfidence": 1,  
                "URL": "string",  
                "ExternalImageId": "string",  
                "ExternalImageUrl": "string",  
                "ExternalImageS3Url": "string",  
                "ExternalImageS3Object": {  
                    "Bucket": "string",  
                    "Name": "string",  
                    "Version": "string"  
                }  
            },  
            "Face": {  
                "BoundingBox": {  
                    "Width": 1,  
                    "Height": 1,  
                    "Left": 1,  
                    "Top": 1  
                },  
                "Angle": 1,  
                "Landmarks": [  
                    {  
                        "Name": "string",  
                        "X": 1,  
                        "Y": 1  
                    }  
                ],  
                "Pose": {  
                    "Roll": 1,  
                    "Pitch": 1,  
                    "Yaw": 1  
                },  
                "Quality": 1  
            }  
        }  
    ]  
}
```

```
"Face": {  
    "BoundingBox": {  
        "Height": number,  
        "Left": number,  
        "Top": number,  
        "Width": number  
    },  
    "Confidence": number,  
    "Landmarks": [  
        {  
            "Type": "string",  
            "X": number,  
            "Y": number  
        }  
    ],  
    "Pose": {  
        "Pitch": number,  
        "Roll": number,  
        "Yaw": number  
    },  
    "Quality": {  
        "Brightness": number,  
        "Sharpness": number  
    }  
},  
    "Id": "string",  
    "MatchConfidence": number,  
    "Name": "string",  
    "Urls": [ "string" ]  
},  
],  
"OrientationCorrection": "string",  
"UnrecognizedFaces": [  
    {  

```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

CelebrityFaces (p. 304)

Details about each celebrity found in the image. Amazon Rekognition can detect a maximum of 15 celebrities in an image.

Type: Array of [Celebrity \(p. 346\)](#) objects

OrientationCorrection (p. 304)

The orientation of the input image (counterclockwise direction). If your application displays the image, you can use this value to correct the orientation. The bounding box coordinates returned in `CelebrityFaces` and `UnrecognizedFaces` represent face locations before the image orientation is corrected.

Note

If the input image is in .jpeg format, it might contain exchangeable image (Exif) metadata that includes the image's orientation. If so, and the Exif metadata for the input image populates the orientation field, the value of `OrientationCorrection` is null. The `CelebrityFaces` and `UnrecognizedFaces` bounding box coordinates represent face locations after Exif metadata is used to correct the image orientation. Images in .png format don't contain Exif metadata.

Type: String

Valid Values: `ROTATE_0` | `ROTATE_90` | `ROTATE_180` | `ROTATE_270`

UnrecognizedFaces (p. 304)

Details about each unrecognized face in the image.

Type: Array of [ComparedFace \(p. 350\)](#) objects

Errors

AccessDeniedException

You are not authorized to perform the action.

HTTP Status Code: 400

ImageTooLargeException

The input image size exceeds the allowed limit. For more information, see [Amazon Rekognition의 제한 \(p. 400\)](#).

HTTP Status Code: 400

InternalServerError

Amazon Rekognition experienced a service issue. Try your call again.

HTTP Status Code: 500

InvalidImageFormatException

The provided image format is not supported.

HTTP Status Code: 400

InvalidImageFormatException

The provided image format is not supported.

HTTP Status Code: 400

InvalidParameterException

Input parameter violated a constraint. Validate your parameter before calling the API operation again.

HTTP Status Code: 400

InvalidS3ObjectException

Amazon Rekognition is unable to access the S3 object specified in the request.

HTTP Status Code: 400

ProvisionedThroughputExceededException

The number of requests exceeded your throughput limit. If you want to increase this limit, contact Amazon Rekognition.

HTTP Status Code: 400

ThrottlingException

Amazon Rekognition is temporarily unable to process the request. Try your call again.

HTTP Status Code: 500

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

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

SearchFaces

For a given input face ID, searches for matching faces in the collection the face belongs to. You get a face ID when you add a face to the collection using the [IndexFaces \(p. 287\)](#) operation. The operation compares the features of the input face with faces in the specified collection.

Note

You can also search faces without indexing faces by using the [SearchFacesByImage](#) operation.

The operation response returns an array of faces that match, ordered by similarity score with the highest similarity first. More specifically, it is an array of metadata for each face match that is found. Along with the metadata, the response also includes a confidence value for each face match, indicating the confidence that the specific face matches the input face.

For an example, see [얼굴 ID를 사용하여 얼굴 검색 \(p. 152\)](#).

This operation requires permissions to perform the `rekognition:SearchFaces` action.

Request Syntax

```
{  
    "CollectionId": "string",  
    "FaceId": "string",  
    "FaceMatchThreshold": number,  
    "MaxFaces": number  
}
```

Request Parameters

The request accepts the following data in JSON format.

CollectionId ([p. 308](#))

ID of the collection the face belongs to.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 255.

Pattern: [a-zA-Z0-9_.\-\+]+

Required: Yes

Facelid ([p. 308](#))

ID of a face to find matches for in the collection.

Type: String

Pattern: [0-9a-f]{8}-[0-9a-f]{4}-[0-9a-f]{4}-[0-9a-f]{4}-[0-9a-f]{12}

Required: Yes

FaceMatchThreshold ([p. 308](#))

Optional value specifying the minimum confidence in the face match to return. For example, don't return any matches where confidence in matches is less than 70%.

Type: Float

Valid Range: Minimum value of 0. Maximum value of 100.

Required: No

[MaxFaces \(p. 308\)](#)

Maximum number of faces to return. The operation returns the maximum number of faces with the highest confidence in the match.

Type: Integer

Valid Range: Minimum value of 1. Maximum value of 4096.

Required: No

Response Syntax

```
{  
    "FaceMatches": [  
        {  
            "Face": {  
                "BoundingBox": {  
                    "Height": "number",  
                    "Left": "number",  
                    "Top": "number",  
                    "Width": "number"  
                },  
                "Confidence": "number",  
                "ExternalImageId": "string",  
                "FaceId": "string",  
                "ImageId": "string"  
            },  
            "Similarity": "number"  
        }  
    ],  
    "FaceModelVersion": "string",  
    "SearchedFaceId": "string"  
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

[FaceMatches \(p. 309\)](#)

An array of faces that matched the input face, along with the confidence in the match.

Type: Array of [FaceMatch \(p. 363\)](#) objects

[FaceModelVersion \(p. 309\)](#)

Version number of the face detection model associated with the input collection (`CollectionId`).

Type: String

[SearchedFaceld \(p. 309\)](#)

ID of the face that was searched for matches in a collection.

Type: String

Pattern: [0-9a-f]{8}-[0-9a-f]{4}-[0-9a-f]{4}-[0-9a-f]{4}-[0-9a-f]{12}

Errors

AccessDeniedException

You are not authorized to perform the action.

HTTP Status Code: 400

InternalServerError

Amazon Rekognition experienced a service issue. Try your call again.

HTTP Status Code: 500

InvalidParameterException

Input parameter violated a constraint. Validate your parameter before calling the API operation again.

HTTP Status Code: 400

ProvisionedThroughputExceededException

The number of requests exceeded your throughput limit. If you want to increase this limit, contact Amazon Rekognition.

HTTP Status Code: 400

ResourceNotFoundException

The collection specified in the request cannot be found.

HTTP Status Code: 400

ThrottlingException

Amazon Rekognition is temporarily unable to process the request. Try your call again.

HTTP Status Code: 500

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

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

SearchFacesByImage

For a given input image, first detects the largest face in the image, and then searches the specified collection for matching faces. The operation compares the features of the input face with faces in the specified collection.

Note

To search for all faces in an input image, you might first call the [IndexFaces \(p. 287\)](#) operation, and then use the face IDs returned in subsequent calls to the [SearchFaces \(p. 308\)](#) operation. You can also call the [DetectFaces](#) operation and use the bounding boxes in the response to make face crops, which then you can pass in to the [SearchFacesByImage](#) operation.

You pass the input image either as base64-encoded image bytes or as a reference to an image in an Amazon S3 bucket. If you use the AWS CLI to call Amazon Rekognition operations, passing image bytes is not supported. The image must be either a PNG or JPEG formatted file.

The response returns an array of faces that match, ordered by similarity score with the highest similarity first. More specifically, it is an array of metadata for each face match found. Along with the metadata, the response also includes a `similarity` indicating how similar the face is to the input face. In the response, the operation also returns the bounding box (and a confidence level that the bounding box contains a face) of the face that Amazon Rekognition used for the input image.

For an example, see [이미지를 사용하여 얼굴 검색 \(p. 156\)](#).

This operation requires permissions to perform the `rekognition:SearchFacesByImage` action.

Request Syntax

```
{  
    "CollectionId": "string",  
    "FaceMatchThreshold": number,  
    "Image": {  
        "Bytes": blob,  
        "S3Object": {  
            "Bucket": "string",  
            "Name": "string",  
            "Version": "string"  
        }  
    },  
    "MaxFaces": number  
}
```

Request Parameters

The request accepts the following data in JSON format.

CollectionId (p. 311)

ID of the collection to search.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 255.

Pattern: [a-zA-Z0-9_.\-\+]

Required: Yes

[FaceMatchThreshold \(p. 311\)](#)

(Optional) Specifies the minimum confidence in the face match to return. For example, don't return any matches where confidence in matches is less than 70%.

Type: Float

Valid Range: Minimum value of 0. Maximum value of 100.

Required: No

[Image \(p. 311\)](#)

The input image as base64-encoded bytes or an S3 object. If you use the AWS CLI to call Amazon Rekognition operations, passing base64-encoded image bytes is not supported.

Type: [Image \(p. 368\)](#) object

Required: Yes

[MaxFaces \(p. 311\)](#)

Maximum number of faces to return. The operation returns the maximum number of faces with the highest confidence in the match.

Type: Integer

Valid Range: Minimum value of 1. Maximum value of 4096.

Required: No

Response Syntax

```
{  
    "FaceMatches": [  
        {  
            "Face": {  
                "BoundingBox": {  
                    "Height": number,  
                    "Left": number,  
                    "Top": number,  
                    "Width": number  
                },  
                "Confidence": number,  
                "ExternalImageId": "string",  
                "FaceId": "string",  
                "ImageId": "string"  
            },  
            "Similarity": number  
        }  
    ],  
    "FaceModelVersion": "string",  
    "SearchedFaceBoundingBox": {  
        "Height": number,  
        "Left": number,  
        "Top": number,  
        "Width": number  
    },  
    "SearchedFaceConfidence": number  
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

[FaceMatches \(p. 312\)](#)

An array of faces that match the input face, along with the confidence in the match.

Type: Array of [FaceMatch \(p. 363\)](#) objects

[FaceModelVersion \(p. 312\)](#)

Version number of the face detection model associated with the input collection (`CollectionId`).

Type: String

[SearchedFaceBoundingBox \(p. 312\)](#)

The bounding box around the face in the input image that Amazon Rekognition used for the search.

Type: [BoundingBox \(p. 344\)](#) object

[SearchedFaceConfidence \(p. 312\)](#)

The level of confidence that the `searchedFaceBoundingBox` contains a face.

Type: Float

Valid Range: Minimum value of 0. Maximum value of 100.

Errors

[AccessDeniedException](#)

You are not authorized to perform the action.

HTTP Status Code: 400

[ImageTooLargeException](#)

The input image size exceeds the allowed limit. For more information, see [Amazon Rekognition의 제한 \(p. 400\)](#).

HTTP Status Code: 400

[InternalServerError](#)

Amazon Rekognition experienced a service issue. Try your call again.

HTTP Status Code: 500

[InvalidImageFormatException](#)

The provided image format is not supported.

HTTP Status Code: 400

[InvalidParameterException](#)

Input parameter violated a constraint. Validate your parameter before calling the API operation again.

HTTP Status Code: 400

InvalidS3ObjectException

Amazon Rekognition is unable to access the S3 object specified in the request.

HTTP Status Code: 400

ProvisionedThroughputExceededException

The number of requests exceeded your throughput limit. If you want to increase this limit, contact Amazon Rekognition.

HTTP Status Code: 400

ResourceNotFoundException

The collection specified in the request cannot be found.

HTTP Status Code: 400

ThrottlingException

Amazon Rekognition is temporarily unable to process the request. Try your call again.

HTTP Status Code: 500

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

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

StartCelebrityRecognition

Starts asynchronous recognition of celebrities in a stored video.

Amazon Rekognition Video can detect celebrities in a video must be stored in an Amazon S3 bucket. Use [Video \(p. 397\)](#) to specify the bucket name and the filename of the video.

`StartCelebrityRecognition` returns a job identifier (`JobId`) which you use to get the results of the analysis. When celebrity recognition analysis is finished, Amazon Rekognition Video publishes a completion status to the Amazon Simple Notification Service topic that you specify in `NotificationChannel`. To get the results of the celebrity recognition analysis, first check that the status value published to the Amazon SNS topic is `SUCCEEDED`. If so, call [GetCelebrityRecognition \(p. 259\)](#) and pass the job identifier (`JobId`) from the initial call to `StartCelebrityRecognition`.

For more information, see [유명 인사 인식 \(p. 173\)](#).

Request Syntax

```
{  
    "ClientRequestToken": "string",  
    "JobTag": "string",  
    "NotificationChannel": {  
        "RoleArn": "string",  
        "SNSTopicArn": "string"  
    },  
    "Video": {  
        "S3Object": {  
            "Bucket": "string",  
            "Name": "string",  
            "Version": "string"  
        }  
    }  
}
```

Request Parameters

The request accepts the following data in JSON format.

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

Idempotent token used to identify the start request. If you use the same token with multiple `StartCelebrityRecognition` requests, the same `JobId` is returned. Use `ClientRequestToken` to prevent the same job from being accidentally started more than once.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 64.

Pattern: ^[a-zA-Z0-9-_]+\$

Required: No

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

Unique identifier you specify to identify the job in the completion status published to the Amazon Simple Notification Service topic.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 256.

Pattern: [a-zA-Z0-9_.\-\:]+

Required: No

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

The Amazon SNS topic ARN that you want Amazon Rekognition Video to publish the completion status of the celebrity recognition analysis to.

Type: [NotificationChannel \(p. 380\)](#) object

Required: No

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

The video in which you want to recognize celebrities. The video must be stored in an Amazon S3 bucket.

Type: [Video \(p. 397\)](#) object

Required: Yes

Response Syntax

```
{  
    "JobId": "string"  
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

[JobId \(p. 316\)](#)

The identifier for the celebrity recognition analysis job. Use `JobId` to identify the job in a subsequent call to `GetCelebrityRecognition`.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 64.

Pattern: ^[a-zA-Z0-9-_]+\$

Errors

AccessDeniedException

You are not authorized to perform the action.

HTTP Status Code: 400

IdempotentParameterMismatchException

A `ClientRequestToken` input parameter was reused with an operation, but at least one of the other input parameters is different from the previous call to the operation.

HTTP Status Code: 400

InternalServerError

Amazon Rekognition experienced a service issue. Try your call again.

HTTP Status Code: 500

InvalidParameterException

Input parameter violated a constraint. Validate your parameter before calling the API operation again.

HTTP Status Code: 400

InvalidS3ObjectException

Amazon Rekognition is unable to access the S3 object specified in the request.

HTTP Status Code: 400

LimitExceededException

An Amazon Rekognition service limit was exceeded. For example, if you start too many Amazon Rekognition Video jobs concurrently, calls to start operations (`StartLabelDetection`, for example) will raise a `LimitExceededException` exception (HTTP status code: 400) until the number of concurrently running jobs is below the Amazon Rekognition service limit.

HTTP Status Code: 400

ProvisionedThroughputExceededException

The number of requests exceeded your throughput limit. If you want to increase this limit, contact Amazon Rekognition.

HTTP Status Code: 400

ThrottlingException

Amazon Rekognition is temporarily unable to process the request. Try your call again.

HTTP Status Code: 500

VideoTooLargeException

The file size or duration of the supplied media is too large. The maximum file size is 8GB. The maximum duration is 2 hours.

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

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

StartContentModeration

Starts asynchronous detection of explicit or suggestive adult content in a stored video.

Amazon Rekognition Video can moderate content in a video stored in an Amazon S3 bucket. Use [Video \(p. 397\)](#) to specify the bucket name and the filename of the video. `StartContentModeration` returns a job identifier (`JobId`) which you use to get the results of the analysis. When content moderation analysis is finished, Amazon Rekognition Video publishes a completion status to the Amazon Simple Notification Service topic that you specify in `NotificationChannel`.

To get the results of the content moderation analysis, first check that the status value published to the Amazon SNS topic is `SUCCEEDED`. If so, call [GetContentModeration \(p. 264\)](#) and pass the job identifier (`JobId`) from the initial call to `StartContentModeration`.

For more information, see [비안전 콘텐츠 감지 \(p. 187\)](#).

Request Syntax

```
{  
    "ClientRequestToken": "string",  
    "JobTag": "string",  
    "MinConfidence": number,  
    "NotificationChannel": {  
        "RoleArn": "string",  
        "SNSTopicArn": "string"  
    },  
    "Video": {  
        "S3Object": {  
            "Bucket": "string",  
            "Name": "string",  
            "Version": "string"  
        }  
    }  
}
```

Request Parameters

The request accepts the following data in JSON format.

[ClientRequestToken \(p. 318\)](#)

Idempotent token used to identify the start request. If you use the same token with multiple `StartContentModeration` requests, the same `JobId` is returned. Use `ClientRequestToken` to prevent the same job from being accidentally started more than once.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 64.

Pattern: ^[a-zA-Z0-9-_]+\$

Required: No

[JobTag \(p. 318\)](#)

Unique identifier you specify to identify the job in the completion status published to the Amazon Simple Notification Service topic.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 256.

Pattern: [a-zA-Z0-9_.\-\:]⁺

Required: No

[MinConfidence \(p. 318\)](#)

Specifies the minimum confidence that Amazon Rekognition must have in order to return a moderated content label. Confidence represents how certain Amazon Rekognition is that the moderated content is correctly identified. 0 is the lowest confidence. 100 is the highest confidence. Amazon Rekognition doesn't return any moderated content labels with a confidence level lower than this specified value. If you don't specify `MinConfidence`, `GetContentModeration` returns labels with confidence values greater than or equal to 50 percent.

Type: Float

Valid Range: Minimum value of 0. Maximum value of 100.

Required: No

[NotificationChannel \(p. 318\)](#)

The Amazon SNS topic ARN that you want Amazon Rekognition Video to publish the completion status of the content moderation analysis to.

Type: [NotificationChannel \(p. 380\)](#) object

Required: No

[Video \(p. 318\)](#)

The video in which you want to moderate content. The video must be stored in an Amazon S3 bucket.

Type: [Video \(p. 397\)](#) object

Required: Yes

Response Syntax

```
{  
    "JobId": "string"  
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

[JobId \(p. 319\)](#)

The identifier for the content moderation analysis job. Use `JobId` to identify the job in a subsequent call to `GetContentModeration`.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 64.

Pattern: ^[a-zA-Z0-9-_]+\$

Errors

AccessDeniedException

You are not authorized to perform the action.

HTTP Status Code: 400

IdempotentParameterMismatchException

A `ClientRequestToken` input parameter was reused with an operation, but at least one of the other input parameters is different from the previous call to the operation.

HTTP Status Code: 400

InternalServerError

Amazon Rekognition experienced a service issue. Try your call again.

HTTP Status Code: 500

InvalidParameterException

Input parameter violated a constraint. Validate your parameter before calling the API operation again.

HTTP Status Code: 400

InvalidS3ObjectException

Amazon Rekognition is unable to access the S3 object specified in the request.

HTTP Status Code: 400

LimitExceededException

An Amazon Rekognition service limit was exceeded. For example, if you start too many Amazon Rekognition Video jobs concurrently, calls to start operations (`StartLabelDetection`, for example) will raise a `LimitExceededException` exception (HTTP status code: 400) until the number of concurrently running jobs is below the Amazon Rekognition service limit.

HTTP Status Code: 400

ProvisionedThroughputExceededException

The number of requests exceeded your throughput limit. If you want to increase this limit, contact Amazon Rekognition.

HTTP Status Code: 400

ThrottlingException

Amazon Rekognition is temporarily unable to process the request. Try your call again.

HTTP Status Code: 500

VideoTooLargeException

The file size or duration of the supplied media is too large. The maximum file size is 8GB. The maximum duration is 2 hours.

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V2

StartFaceDetection

Starts asynchronous detection of faces in a stored video.

Amazon Rekognition Video can detect faces in a video stored in an Amazon S3 bucket. Use [Video \(p. 397\)](#) to specify the bucket name and the filename of the video. `StartFaceDetection` returns a job identifier (`JobId`) that you use to get the results of the operation. When face detection is finished, Amazon Rekognition Video publishes a completion status to the Amazon Simple Notification Service topic that you specify in `NotificationChannel`. To get the results of the face detection operation, first check that the status value published to the Amazon SNS topic is `SUCCEEDED`. If so, call [GetFaceDetection \(p. 268\)](#) and pass the job identifier (`JobId`) from the initial call to `StartFaceDetection`.

For more information, see [저장된 비디오에서 얼굴 감지 \(p. 121\)](#).

Request Syntax

```
{  
    "ClientRequestToken": "string",  
    "FaceAttributes": "string",  
    "JobTag": "string",  
    "NotificationChannel": {  
        "RoleArn": "string",  
        "SNSTopicArn": "string"  
    },  
    "Video": {  
        "S3Object": {  
            "Bucket": "string",  
            "Name": "string",  
            "Version": "string"  
        }  
    }  
}
```

Request Parameters

The request accepts the following data in JSON format.

[ClientRequestToken \(p. 322\)](#)

Idempotent token used to identify the start request. If you use the same token with multiple `StartFaceDetection` requests, the same `JobId` is returned. Use `ClientRequestToken` to prevent the same job from being accidentally started more than once.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 64.

Pattern: ^[a-zA-Z0-9-_]+\$

Required: No

[FaceAttributes \(p. 322\)](#)

The face attributes you want returned.

`DEFAULT` - The following subset of facial attributes are returned: `BoundingBox`, `Confidence`, `Pose`, `Quality` and `Landmarks`.

ALL - All facial attributes are returned.

Type: String

Valid Values: `DEFAULT` | `ALL`

Required: No

[JobTag \(p. 322\)](#)

Unique identifier you specify to identify the job in the completion status published to the Amazon Simple Notification Service topic.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 256.

Pattern: `[a-zA-Z0-9_.\-\:]+`

Required: No

[NotificationChannel \(p. 322\)](#)

The ARN of the Amazon SNS topic to which you want Amazon Rekognition Video to publish the completion status of the face detection operation.

Type: [NotificationChannel \(p. 380\)](#) object

Required: No

[Video \(p. 322\)](#)

The video in which you want to detect faces. The video must be stored in an Amazon S3 bucket.

Type: [Video \(p. 397\)](#) object

Required: Yes

Response Syntax

```
{  
    "JobId": "string"  
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

[JobId \(p. 323\)](#)

The identifier for the face detection job. Use `JobId` to identify the job in a subsequent call to `GetFaceDetection`.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 64.

Pattern: `^[a-zA-Z0-9-_]+$`

Errors

AccessDeniedException

You are not authorized to perform the action.

HTTP Status Code: 400

IdempotentParameterMismatchException

A `ClientRequestToken` input parameter was reused with an operation, but at least one of the other input parameters is different from the previous call to the operation.

HTTP Status Code: 400

InternalServerError

Amazon Rekognition experienced a service issue. Try your call again.

HTTP Status Code: 500

InvalidParameterException

Input parameter violated a constraint. Validate your parameter before calling the API operation again.

HTTP Status Code: 400

InvalidS3ObjectException

Amazon Rekognition is unable to access the S3 object specified in the request.

HTTP Status Code: 400

LimitExceededException

An Amazon Rekognition service limit was exceeded. For example, if you start too many Amazon Rekognition Video jobs concurrently, calls to start operations (`StartLabelDetection`, for example) will raise a `LimitExceededException` exception (HTTP status code: 400) until the number of concurrently running jobs is below the Amazon Rekognition service limit.

HTTP Status Code: 400

ProvisionedThroughputExceededException

The number of requests exceeded your throughput limit. If you want to increase this limit, contact Amazon Rekognition.

HTTP Status Code: 400

ThrottlingException

Amazon Rekognition is temporarily unable to process the request. Try your call again.

HTTP Status Code: 500

VideoTooLargeException

The file size or duration of the supplied media is too large. The maximum file size is 8GB. The maximum duration is 2 hours.

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V2

StartFaceSearch

Starts the asynchronous search for faces in a collection that match the faces of persons detected in a stored video.

The video must be stored in an Amazon S3 bucket. Use [Video \(p. 397\)](#) to specify the bucket name and the filename of the video. StartFaceSearch returns a job identifier (JobId) which you use to get the search results once the search has completed. When searching is finished, Amazon Rekognition Video publishes a completion status to the Amazon Simple Notification Service topic that you specify in `NotificationChannel`. To get the search results, first check that the status value published to the Amazon SNS topic is `SUCCEEDED`. If so, call [GetFaceSearch \(p. 273\)](#) and pass the job identifier (JobId) from the initial call to `StartFaceSearch`. For more information, see [저장된 비디오에서 얼굴 검색 \(p. 160\)](#).

Request Syntax

```
{  
    "ClientRequestToken": "string",  
    "CollectionId": "string",  
    "FaceMatchThreshold": number,  
    "JobTag": "string",  
    "NotificationChannel": {  
        "RoleArn": "string",  
        "SNSTopicArn": "string"  
    },  
    "Video": {  
        "S3Object": {  
            "Bucket": "string",  
            "Name": "string",  
            "Version": "string"  
        }  
    }  
}
```

Request Parameters

The request accepts the following data in JSON format.

[ClientRequestToken \(p. 326\)](#)

Idempotent token used to identify the start request. If you use the same token with multiple `StartFaceSearch` requests, the same `JobId` is returned. Use `ClientRequestToken` to prevent the same job from being accidentally started more than once.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 64.

Pattern: ^[a-zA-Z0-9-_]+\$

Required: No

[CollectionId \(p. 326\)](#)

ID of the collection that contains the faces you want to search for.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 255.

Pattern: [a-zA-Z0-9_.\-_]+

Required: Yes

[FaceMatchThreshold \(p. 326\)](#)

The minimum confidence in the person match to return. For example, don't return any matches where confidence in matches is less than 70%.

Type: Float

Valid Range: Minimum value of 0. Maximum value of 100.

Required: No

[JobTag \(p. 326\)](#)

Unique identifier you specify to identify the job in the completion status published to the Amazon Simple Notification Service topic.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 256.

Pattern: [a-zA-Z0-9_.\-_]+

Required: No

[NotificationChannel \(p. 326\)](#)

The ARN of the Amazon SNS topic to which you want Amazon Rekognition Video to publish the completion status of the search.

Type: [NotificationChannel \(p. 380\)](#) object

Required: No

[Video \(p. 326\)](#)

The video you want to search. The video must be stored in an Amazon S3 bucket.

Type: [Video \(p. 397\)](#) object

Required: Yes

Response Syntax

```
{  
    "JobId": "string"  
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

[JobId \(p. 327\)](#)

The identifier for the search job. Use `JobId` to identify the job in a subsequent call to `GetFaceSearch`.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 64.

Pattern: ^[a-zA-Z0-9-_]+\$

Errors

AccessDeniedException

You are not authorized to perform the action.

HTTP Status Code: 400

IdempotentParameterMismatchException

A `ClientRequestToken` input parameter was reused with an operation, but at least one of the other input parameters is different from the previous call to the operation.

HTTP Status Code: 400

InternalServerError

Amazon Rekognition experienced a service issue. Try your call again.

HTTP Status Code: 500

InvalidParameterException

Input parameter violated a constraint. Validate your parameter before calling the API operation again.

HTTP Status Code: 400

InvalidS3ObjectException

Amazon Rekognition is unable to access the S3 object specified in the request.

HTTP Status Code: 400

LimitExceededException

An Amazon Rekognition service limit was exceeded. For example, if you start too many Amazon Rekognition Video jobs concurrently, calls to start operations (`StartLabelDetection`, for example) will raise a `LimitExceededException` exception (HTTP status code: 400) until the number of concurrently running jobs is below the Amazon Rekognition service limit.

HTTP Status Code: 400

ProvisionedThroughputExceededException

The number of requests exceeded your throughput limit. If you want to increase this limit, contact Amazon Rekognition.

HTTP Status Code: 400

ResourceNotFoundException

The collection specified in the request cannot be found.

HTTP Status Code: 400

ThrottlingException

Amazon Rekognition is temporarily unable to process the request. Try your call again.

HTTP Status Code: 500

VideoTooLargeException

The file size or duration of the supplied media is too large. The maximum file size is 8GB. The maximum duration is 2 hours.

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

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

StartLabelDetection

Starts asynchronous detection of labels in a stored video.

Amazon Rekognition Video can detect labels in a video. Labels are instances of real-world entities. This includes objects like flower, tree, and table; events like wedding, graduation, and birthday party; concepts like landscape, evening, and nature; and activities like a person getting out of a car or a person skiing.

The video must be stored in an Amazon S3 bucket. Use [Video \(p. 397\)](#) to specify the bucket name and the filename of the video. `StartLabelDetection` returns a job identifier (`JobId`) which you use to get the results of the operation. When label detection is finished, Amazon Rekognition Video publishes a completion status to the Amazon Simple Notification Service topic that you specify in `NotificationChannel`.

To get the results of the label detection operation, first check that the status value published to the Amazon SNS topic is `SUCCEEDED`. If so, call [GetLabelDetection \(p. 278\)](#) and pass the job identifier (`JobId`) from the initial call to `StartLabelDetection`.

Request Syntax

```
{  
    "ClientRequestToken": "string",  
    "JobTag": "string",  
    "MinConfidence": number,  
    "NotificationChannel": {  
        "RoleArn": "string",  
        "SNSTopicArn": "string"  
    },  
    "Video": {  
        "S3Object": {  
            "Bucket": "string",  
            "Name": "string",  
            "Version": "string"  
        }  
    }  
}
```

Request Parameters

The request accepts the following data in JSON format.

[ClientRequestToken \(p. 330\)](#)

Idempotent token used to identify the start request. If you use the same token with multiple `StartLabelDetection` requests, the same `JobId` is returned. Use `ClientRequestToken` to prevent the same job from being accidentally started more than once.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 64.

Pattern: ^[a-zA-Z0-9-_]+\$

Required: No

[JobTag \(p. 330\)](#)

Unique identifier you specify to identify the job in the completion status published to the Amazon Simple Notification Service topic.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 256.

Pattern: [a-zA-Z0-9_.\-\:]+

Required: No

[MinConfidence \(p. 330\)](#)

Specifies the minimum confidence that Amazon Rekognition Video must have in order to return a detected label. Confidence represents how certain Amazon Rekognition is that a label is correctly identified. 0 is the lowest confidence. 100 is the highest confidence. Amazon Rekognition Video doesn't return any labels with a confidence level lower than this specified value.

If you don't specify `MinConfidence`, the operation returns labels with confidence values greater than or equal to 50 percent.

Type: Float

Valid Range: Minimum value of 0. Maximum value of 100.

Required: No

[NotificationChannel \(p. 330\)](#)

The Amazon SNS topic ARN you want Amazon Rekognition Video to publish the completion status of the label detection operation to.

Type: [NotificationChannel \(p. 380\)](#) object

Required: No

[Video \(p. 330\)](#)

The video in which you want to detect labels. The video must be stored in an Amazon S3 bucket.

Type: [Video \(p. 397\)](#) object

Required: Yes

Response Syntax

```
{  
    "JobId": "string"  
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

[JobId \(p. 331\)](#)

The identifier for the label detection job. Use `JobID` to identify the job in a subsequent call to `GetLabelDetection`.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 64.

Pattern: ^[a-zA-Z0-9-_]+\$

Errors

AccessDeniedException

You are not authorized to perform the action.

HTTP Status Code: 400

IdempotentParameterMismatchException

A `ClientRequestToken` input parameter was reused with an operation, but at least one of the other input parameters is different from the previous call to the operation.

HTTP Status Code: 400

InternalServerError

Amazon Rekognition experienced a service issue. Try your call again.

HTTP Status Code: 500

InvalidParameterException

Input parameter violated a constraint. Validate your parameter before calling the API operation again.

HTTP Status Code: 400

InvalidS3ObjectException

Amazon Rekognition is unable to access the S3 object specified in the request.

HTTP Status Code: 400

LimitExceededException

An Amazon Rekognition service limit was exceeded. For example, if you start too many Amazon Rekognition Video jobs concurrently, calls to start operations (`StartLabelDetection`, for example) will raise a `LimitExceededException` exception (HTTP status code: 400) until the number of concurrently running jobs is below the Amazon Rekognition service limit.

HTTP Status Code: 400

ProvisionedThroughputExceededException

The number of requests exceeded your throughput limit. If you want to increase this limit, contact Amazon Rekognition.

HTTP Status Code: 400

ThrottlingException

Amazon Rekognition is temporarily unable to process the request. Try your call again.

HTTP Status Code: 500

VideoTooLargeException

The file size or duration of the supplied media is too large. The maximum file size is 8GB. The maximum duration is 2 hours.

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

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

StartPersonTracking

Starts the asynchronous tracking of a person's path in a stored video.

Amazon Rekognition Video can track the path of people in a video stored in an Amazon S3 bucket. Use [Video \(p. 397\)](#) to specify the bucket name and the filename of the video. `StartPersonTracking` returns a job identifier (`JobId`) which you use to get the results of the operation. When label detection is finished, Amazon Rekognition publishes a completion status to the Amazon Simple Notification Service topic that you specify in `NotificationChannel`.

To get the results of the person detection operation, first check that the status value published to the Amazon SNS topic is `SUCCEEDED`. If so, call [GetPersonTracking \(p. 282\)](#) and pass the job identifier (`JobId`) from the initial call to `StartPersonTracking`.

Request Syntax

```
{  
    "ClientRequestToken": "string",  
    "JobTag": "string",  
    "NotificationChannel": {  
        "RoleArn": "string",  
        "SNSTopicArn": "string"  
    },  
    "Video": {  
        "S3Object": {  
            "Bucket": "string",  
            "Name": "string",  
            "Version": "string"  
        }  
    }  
}
```

Request Parameters

The request accepts the following data in JSON format.

[ClientRequestToken \(p. 334\)](#)

Idempotent token used to identify the start request. If you use the same token with multiple `StartPersonTracking` requests, the same `JobId` is returned. Use `ClientRequestToken` to prevent the same job from being accidentally started more than once.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 64.

Pattern: `^[a-zA-Z0-9-_]+$`

Required: No

[JobTag \(p. 334\)](#)

Unique identifier you specify to identify the job in the completion status published to the Amazon Simple Notification Service topic.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 256.

Pattern: `[a-zA-Z0-9_.\-\:]+`

Required: No

[NotificationChannel \(p. 334\)](#)

The Amazon SNS topic ARN you want Amazon Rekognition Video to publish the completion status of the people detection operation to.

Type: [NotificationChannel \(p. 380\)](#) object

Required: No

[Video \(p. 334\)](#)

The video in which you want to detect people. The video must be stored in an Amazon S3 bucket.

Type: [Video \(p. 397\)](#) object

Required: Yes

Response Syntax

```
{  
    "JobId": "string"  
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

[JobId \(p. 335\)](#)

The identifier for the person detection job. Use `JobId` to identify the job in a subsequent call to `GetPersonTracking`.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 64.

Pattern: `^[a-zA-Z0-9-_]+$`

Errors

[AccessDeniedException](#)

You are not authorized to perform the action.

HTTP Status Code: 400

[IdempotentParameterMismatchException](#)

A `ClientRequestToken` input parameter was reused with an operation, but at least one of the other input parameters is different from the previous call to the operation.

HTTP Status Code: 400

[InternalServerError](#)

Amazon Rekognition experienced a service issue. Try your call again.

HTTP Status Code: 500

InvalidOperationException

Input parameter violated a constraint. Validate your parameter before calling the API operation again.

HTTP Status Code: 400

InvalidS3ObjectException

Amazon Rekognition is unable to access the S3 object specified in the request.

HTTP Status Code: 400

LimitExceededException

An Amazon Rekognition service limit was exceeded. For example, if you start too many Amazon Rekognition Video jobs concurrently, calls to start operations (`StartLabelDetection`, for example) will raise a `LimitExceededException` exception (HTTP status code: 400) until the number of concurrently running jobs is below the Amazon Rekognition service limit.

HTTP Status Code: 400

ProvisionedThroughputExceededException

The number of requests exceeded your throughput limit. If you want to increase this limit, contact Amazon Rekognition.

HTTP Status Code: 400

ThrottlingException

Amazon Rekognition is temporarily unable to process the request. Try your call again.

HTTP Status Code: 500

VideoTooLargeException

The file size or duration of the supplied media is too large. The maximum file size is 8GB. The maximum duration is 2 hours.

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

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

StartStreamProcessor

Starts processing a stream processor. You create a stream processor by calling [CreateStreamProcessor \(p. 227\)](#). To tell `StartStreamProcessor` which stream processor to start, use the value of the `Name` field specified in the call to `CreateStreamProcessor`.

Request Syntax

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

Request Parameters

The request accepts the following data in JSON format.

Name (p. 337)

The name of the stream processor to start processing.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 128.

Pattern: [a-zA-Z0-9_.\-\-]+

Required: Yes

Response Elements

If the action is successful, the service sends back an HTTP 200 response with an empty HTTP body.

Errors

AccessDeniedException

You are not authorized to perform the action.

HTTP Status Code: 400

InternalServerError

Amazon Rekognition experienced a service issue. Try your call again.

HTTP Status Code: 500

InvalidArgumentException

Input parameter violated a constraint. Validate your parameter before calling the API operation again.

HTTP Status Code: 400

ProvisionedThroughputExceededException

The number of requests exceeded your throughput limit. If you want to increase this limit, contact Amazon Rekognition.

HTTP Status Code: 400

ResourceInUseException

HTTP Status Code: 400

ResourceNotFoundException

The collection specified in the request cannot be found.

HTTP Status Code: 400

ThrottlingException

Amazon Rekognition is temporarily unable to process the request. Try your call again.

HTTP Status Code: 500

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

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

StopStreamProcessor

Stops a running stream processor that was created by [CreateStreamProcessor \(p. 227\)](#).

Request Syntax

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

Request Parameters

The request accepts the following data in JSON format.

Name (p. 339)

The name of a stream processor created by [CreateStreamProcessor \(p. 227\)](#).

Type: String

Length Constraints: Minimum length of 1. Maximum length of 128.

Pattern: [a-zA-Z0-9_.\-\+]+

Required: Yes

Response Elements

If the action is successful, the service sends back an HTTP 200 response with an empty HTTP body.

Errors

AccessDeniedException

You are not authorized to perform the action.

HTTP Status Code: 400

InternalServerError

Amazon Rekognition experienced a service issue. Try your call again.

HTTP Status Code: 500

InvalidParameterException

Input parameter violated a constraint. Validate your parameter before calling the API operation again.

HTTP Status Code: 400

ProvisionedThroughputExceededException

The number of requests exceeded your throughput limit. If you want to increase this limit, contact Amazon Rekognition.

HTTP Status Code: 400

ResourceInUseException

HTTP Status Code: 400
ResourceNotFoundException

The collection specified in the request cannot be found.

HTTP Status Code: 400
ThrottlingException

Amazon Rekognition is temporarily unable to process the request. Try your call again.

HTTP Status Code: 500

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

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

Data Types

The following data types are supported:

- [AgeRange \(p. 342\)](#)
- [Beard \(p. 343\)](#)
- [BoundingBox \(p. 344\)](#)
- [Celebrity \(p. 346\)](#)
- [CelebrityDetail \(p. 347\)](#)
- [CelebrityRecognition \(p. 349\)](#)
- [ComparedFace \(p. 350\)](#)
- [ComparedSourceImageFace \(p. 351\)](#)
- [CompareFacesMatch \(p. 352\)](#)
- [ContentModerationDetection \(p. 353\)](#)
- [Emotion \(p. 354\)](#)
- [Eyeglasses \(p. 355\)](#)
- [EyeOpen \(p. 356\)](#)
- [Face \(p. 357\)](#)
- [FaceDetail \(p. 359\)](#)
- [FaceDetection \(p. 362\)](#)
- [FaceMatch \(p. 363\)](#)
- [FaceRecord \(p. 364\)](#)

- [FaceSearchSettings \(p. 365\)](#)
- [Gender \(p. 366\)](#)
- [Geometry \(p. 367\)](#)
- [Image \(p. 368\)](#)
- [ImageQuality \(p. 369\)](#)
- [Instance \(p. 370\)](#)
- [KinesisDataStream \(p. 371\)](#)
- [KinesisVideoStream \(p. 372\)](#)
- [Label \(p. 373\)](#)
- [LabelDetection \(p. 375\)](#)
- [Landmark \(p. 376\)](#)
- [ModerationLabel \(p. 377\)](#)
- [MouthOpen \(p. 378\)](#)
- [Mustache \(p. 379\)](#)
- [NotificationChannel \(p. 380\)](#)
- [Parent \(p. 381\)](#)
- [PersonDetail \(p. 382\)](#)
- [PersonDetection \(p. 383\)](#)
- [PersonMatch \(p. 384\)](#)
- [Point \(p. 385\)](#)
- [Pose \(p. 386\)](#)
- [S3Object \(p. 387\)](#)
- [Smile \(p. 388\)](#)
- [StreamProcessor \(p. 389\)](#)
- [StreamProcessorInput \(p. 390\)](#)
- [StreamProcessorOutput \(p. 391\)](#)
- [StreamProcessorSettings \(p. 392\)](#)
- [Sunglasses \(p. 393\)](#)
- [TextDetection \(p. 394\)](#)
- [UnindexedFace \(p. 396\)](#)
- [Video \(p. 397\)](#)
- [VideoMetadata \(p. 398\)](#)

AgeRange

Structure containing the estimated age range, in years, for a face.

Amazon Rekognition estimates an age range for faces detected in the input image. Estimated age ranges can overlap. A face of a 5-year-old might have an estimated range of 4-6, while the face of a 6-year-old might have an estimated range of 4-8.

Contents

High

The highest estimated age.

Type: Integer

Valid Range: Minimum value of 0.

Required: No

Low

The lowest estimated age.

Type: Integer

Valid Range: Minimum value of 0.

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java](#)
- [AWS SDK for Ruby V2](#)

Beard

Indicates whether or not the face has a beard, and the confidence level in the determination.

Contents

Confidence

Level of confidence in the determination.

Type: Float

Valid Range: Minimum value of 0. Maximum value of 100.

Required: No

Value

Boolean value that indicates whether the face has beard or not.

Type: Boolean

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java](#)
- [AWS SDK for Ruby V2](#)

BoundingBox

Identifies the bounding box around the label, face, or text. The `left` (x-coordinate) and `top` (y-coordinate) are coordinates representing the top and left sides of the bounding box. Note that the upper-left corner of the image is the origin (0,0).

The `top` and `left` values returned are ratios of the overall image size. For example, if the input image is 700x200 pixels, and the top-left coordinate of the bounding box is 350x50 pixels, the API returns a `left` value of 0.5 (350/700) and a `top` value of 0.25 (50/200).

The `width` and `height` values represent the dimensions of the bounding box as a ratio of the overall image dimension. For example, if the input image is 700x200 pixels, and the bounding box width is 70 pixels, the `width` returned is 0.1.

Note

The bounding box coordinates can have negative values. For example, if Amazon Rekognition is able to detect a face that is at the image edge and is only partially visible, the service can return coordinates that are outside the image bounds and, depending on the image edge, you might get negative values or values greater than 1 for the `left` or `top` values.

Contents

Height

Height of the bounding box as a ratio of the overall image height.

Type: Float

Required: No

Left

Left coordinate of the bounding box as a ratio of overall image width.

Type: Float

Required: No

Top

Top coordinate of the bounding box as a ratio of overall image height.

Type: Float

Required: No

Width

Width of the bounding box as a ratio of the overall image width.

Type: Float

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)

- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for Ruby V2

Celebrity

Provides information about a celebrity recognized by the [RecognizeCelebrities \(p. 304\)](#) operation.

Contents

Face

Provides information about the celebrity's face, such as its location on the image.

Type: [ComparedFace \(p. 350\)](#) object

Required: No

Id

A unique identifier for the celebrity.

Type: String

Pattern: [0-9A-Za-z]*

Required: No

MatchConfidence

The confidence, in percentage, that Amazon Rekognition has that the recognized face is the celebrity.

Type: Float

Valid Range: Minimum value of 0. Maximum value of 100.

Required: No

Name

The name of the celebrity.

Type: String

Required: No

Urls

An array of URLs pointing to additional information about the celebrity. If there is no additional information about the celebrity, this list is empty.

Type: Array of strings

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java](#)
- [AWS SDK for Ruby V2](#)

CelebrityDetail

Information about a recognized celebrity.

Contents

BoundingBox

Bounding box around the body of a celebrity.

Type: [BoundingBox \(p. 344\)](#) object

Required: No

Confidence

The confidence, in percentage, that Amazon Rekognition has that the recognized face is the celebrity.

Type: Float

Valid Range: Minimum value of 0. Maximum value of 100.

Required: No

Face

Face details for the recognized celebrity.

Type: [FaceDetail \(p. 359\)](#) object

Required: No

Id

The unique identifier for the celebrity.

Type: String

Pattern: [0-9A-Za-z]*

Required: No

Name

The name of the celebrity.

Type: String

Required: No

Urls

An array of URLs pointing to additional celebrity information.

Type: Array of strings

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)

- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for Ruby V2

CelebrityRecognition

Information about a detected celebrity and the time the celebrity was detected in a stored video. For more information, see [GetCelebrityRecognition \(p. 259\)](#).

Contents

Celebrity

Information about a recognized celebrity.

Type: [CelebrityDetail \(p. 347\)](#) object

Required: No

Timestamp

The time, in milliseconds from the start of the video, that the celebrity was recognized.

Type: Long

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java](#)
- [AWS SDK for Ruby V2](#)

ComparedFace

Provides face metadata for target image faces that are analyzed by `CompareFaces` and `RecognizeCelebrities`.

Contents

BoundingBox

Bounding box of the face.

Type: [BoundingBox \(p. 344\)](#) object

Required: No

Confidence

Level of confidence that what the bounding box contains is a face.

Type: Float

Valid Range: Minimum value of 0. Maximum value of 100.

Required: No

Landmarks

An array of facial landmarks.

Type: Array of [Landmark \(p. 376\)](#) objects

Required: No

Pose

Indicates the pose of the face as determined by its pitch, roll, and yaw.

Type: [Pose \(p. 386\)](#) object

Required: No

Quality

Identifies face image brightness and sharpness.

Type: [ImageQuality \(p. 369\)](#) object

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java](#)
- [AWS SDK for Ruby V2](#)

ComparedSourceImageFace

Type that describes the face Amazon Rekognition chose to compare with the faces in the target. This contains a bounding box for the selected face and confidence level that the bounding box contains a face. Note that Amazon Rekognition selects the largest face in the source image for this comparison.

Contents

BoundingBox

Bounding box of the face.

Type: [BoundingBox \(p. 344\)](#) object

Required: No

Confidence

Confidence level that the selected bounding box contains a face.

Type: Float

Valid Range: Minimum value of 0. Maximum value of 100.

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java](#)
- [AWS SDK for Ruby V2](#)

CompareFacesMatch

Provides information about a face in a target image that matches the source image face analyzed by `CompareFaces`. The `Face` property contains the bounding box of the face in the target image. The `Similarity` property is the confidence that the source image face matches the face in the bounding box.

Contents

Face

Provides face metadata (bounding box and confidence that the bounding box actually contains a face).

Type: [ComparedFace \(p. 350\)](#) object

Required: No

Similarity

Level of confidence that the faces match.

Type: Float

Valid Range: Minimum value of 0. Maximum value of 100.

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java](#)
- [AWS SDK for Ruby V2](#)

ContentModerationDetection

Information about a moderation label detection in a stored video.

Contents

ModerationLabel

The moderation label detected by in the stored video.

Type: [ModerationLabel \(p. 377\)](#) object

Required: No

Timestamp

Time, in milliseconds from the beginning of the video, that the moderation label was detected.

Type: Long

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java](#)
- [AWS SDK for Ruby V2](#)

Emotion

The emotions detected on the face, and the confidence level in the determination. For example, HAPPY, SAD, and ANGRY.

Contents

Confidence

Level of confidence in the determination.

Type: Float

Valid Range: Minimum value of 0. Maximum value of 100.

Required: No

Type

Type of emotion detected.

Type: String

Valid Values: HAPPY | SAD | ANGRY | CONFUSED | DISGUSTED | SURPRISED | CALM | UNKNOWN

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java](#)
- [AWS SDK for Ruby V2](#)

Eyeglasses

Indicates whether or not the face is wearing eye glasses, and the confidence level in the determination.

Contents

Confidence

Level of confidence in the determination.

Type: Float

Valid Range: Minimum value of 0. Maximum value of 100.

Required: No

Value

Boolean value that indicates whether the face is wearing eye glasses or not.

Type: Boolean

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java](#)
- [AWS SDK for Ruby V2](#)

EyeOpen

Indicates whether or not the eyes on the face are open, and the confidence level in the determination.

Contents

Confidence

Level of confidence in the determination.

Type: Float

Valid Range: Minimum value of 0. Maximum value of 100.

Required: No

Value

Boolean value that indicates whether the eyes on the face are open.

Type: Boolean

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java](#)
- [AWS SDK for Ruby V2](#)

Face

Describes the face properties such as the bounding box, face ID, image ID of the input image, and external image ID that you assigned.

Contents

BoundingBox

Bounding box of the face.

Type: [BoundingBox \(p. 344\)](#) object

Required: No

Confidence

Confidence level that the bounding box contains a face (and not a different object such as a tree).

Type: Float

Valid Range: Minimum value of 0. Maximum value of 100.

Required: No

ExternalImageId

Identifier that you assign to all the faces in the input image.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 255.

Pattern: [a-zA-Z0-9_.\-\:]+

Required: No

FacetId

Unique identifier that Amazon Rekognition assigns to the face.

Type: String

Pattern: [0-9a-f]{8}-[0-9a-f]{4}-[0-9a-f]{4}-[0-9a-f]{4}-[0-9a-f]{12}

Required: No

ImageId

Unique identifier that Amazon Rekognition assigns to the input image.

Type: String

Pattern: [0-9a-f]{8}-[0-9a-f]{4}-[0-9a-f]{4}-[0-9a-f]{4}-[0-9a-f]{12}

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)

- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for Ruby V2

FaceDetail

Structure containing attributes of the face that the algorithm detected.

A FaceDetail object contains either the default facial attributes or all facial attributes. The default attributes are BoundingBox, Confidence, Landmarks, Pose, and Quality.

[GetFaceDetection \(p. 268\)](#) is the only Amazon Rekognition Video stored video operation that can return a FaceDetail object with all attributes. To specify which attributes to return, use the FaceAttributes input parameter for [StartFaceDetection \(p. 322\)](#). The following Amazon Rekognition Video operations return only the default attributes. The corresponding Start operations don't have a FaceAttributes input parameter.

- GetCelebrityRecognition
- GetPersonTracking
- GetFaceSearch

The Amazon Rekognition Image [DetectFaces \(p. 243\)](#) and [IndexFaces \(p. 287\)](#) operations can return all facial attributes. To specify which attributes to return, use the Attributes input parameter for [DetectFaces](#). For [IndexFaces](#), use the DetectAttributes input parameter.

Contents

AgeRange

The estimated age range, in years, for the face. Low represents the lowest estimated age and High represents the highest estimated age.

Type: [AgeRange \(p. 342\)](#) object

Required: No

Beard

Indicates whether or not the face has a beard, and the confidence level in the determination.

Type: [Beard \(p. 343\)](#) object

Required: No

BoundingBox

Bounding box of the face. Default attribute.

Type: [BoundingBox \(p. 344\)](#) object

Required: No

Confidence

Confidence level that the bounding box contains a face (and not a different object such as a tree). Default attribute.

Type: Float

Valid Range: Minimum value of 0. Maximum value of 100.

Required: No

Emotions

The emotions detected on the face, and the confidence level in the determination. For example, HAPPY, SAD, and ANGRY.

Type: Array of [Emotion \(p. 354\)](#) objects

Required: No

Eyeglasses

Indicates whether or not the face is wearing eye glasses, and the confidence level in the determination.

Type: [Eyeglasses \(p. 355\)](#) object

Required: No

EyesOpen

Indicates whether or not the eyes on the face are open, and the confidence level in the determination.

Type: [EyeOpen \(p. 356\)](#) object

Required: No

Gender

Gender of the face and the confidence level in the determination.

Type: [Gender \(p. 366\)](#) object

Required: No

Landmarks

Indicates the location of landmarks on the face. Default attribute.

Type: Array of [Landmark \(p. 376\)](#) objects

Required: No

MouthOpen

Indicates whether or not the mouth on the face is open, and the confidence level in the determination.

Type: [MouthOpen \(p. 378\)](#) object

Required: No

Mustache

Indicates whether or not the face has a mustache, and the confidence level in the determination.

Type: [Mustache \(p. 379\)](#) object

Required: No

Pose

Indicates the pose of the face as determined by its pitch, roll, and yaw. Default attribute.

Type: [Pose \(p. 386\)](#) object

Required: No

Quality

Identifies image brightness and sharpness. Default attribute.

Type: [ImageQuality \(p. 369\)](#) object

Required: No

Smile

Indicates whether or not the face is smiling, and the confidence level in the determination.

Type: [Smile \(p. 388\)](#) object

Required: No

Sunglasses

Indicates whether or not the face is wearing sunglasses, and the confidence level in the determination.

Type: [Sunglasses \(p. 393\)](#) object

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java](#)
- [AWS SDK for Ruby V2](#)

FaceDetection

Information about a face detected in a video analysis request and the time the face was detected in the video.

Contents

Face

The face properties for the detected face.

Type: [FaceDetail \(p. 359\)](#) object

Required: No

Timestamp

Time, in milliseconds from the start of the video, that the face was detected.

Type: Long

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java](#)
- [AWS SDK for Ruby V2](#)

FaceMatch

Provides face metadata. In addition, it also provides the confidence in the match of this face with the input face.

Contents

Face

Describes the face properties such as the bounding box, face ID, image ID of the source image, and external image ID that you assigned.

Type: [Face \(p. 357\)](#) object

Required: No

Similarity

Confidence in the match of this face with the input face.

Type: Float

Valid Range: Minimum value of 0. Maximum value of 100.

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java](#)
- [AWS SDK for Ruby V2](#)

FaceRecord

Object containing both the face metadata (stored in the backend database), and facial attributes that are detected but aren't stored in the database.

Contents

Face

Describes the face properties such as the bounding box, face ID, image ID of the input image, and external image ID that you assigned.

Type: [Face \(p. 357\)](#) object

Required: No

FaceDetail

Structure containing attributes of the face that the algorithm detected.

Type: [FaceDetail \(p. 359\)](#) object

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java](#)
- [AWS SDK for Ruby V2](#)

FaceSearchSettings

Input face recognition parameters for an Amazon Rekognition stream processor.
`FaceRecognitionSettings` is a request parameter for [CreateStreamProcessor \(p. 227\)](#).

Contents

CollectionId

The ID of a collection that contains faces that you want to search for.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 255.

Pattern: [a-zA-Z0-9_.\-_]+

Required: No

FaceMatchThreshold

Minimum face match confidence score that must be met to return a result for a recognized face. Default is 70. 0 is the lowest confidence. 100 is the highest confidence.

Type: Float

Valid Range: Minimum value of 0. Maximum value of 100.

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java](#)
- [AWS SDK for Ruby V2](#)

Gender

Gender of the face and the confidence level in the determination.

Contents

Confidence

Level of confidence in the determination.

Type: Float

Valid Range: Minimum value of 0. Maximum value of 100.

Required: No

Value

Gender of the face.

Type: String

Valid Values: `Male` | `Female`

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java](#)
- [AWS SDK for Ruby V2](#)

Geometry

Information about where the text detected by [DetectText \(p. 254\)](#) is located on an image.

Contents

BoundingBox

An axis-aligned coarse representation of the detected text's location on the image.

Type: [BoundingBox \(p. 344\)](#) object

Required: No

Polygon

Within the bounding box, a fine-grained polygon around the detected text.

Type: Array of [Point \(p. 385\)](#) objects

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java](#)
- [AWS SDK for Ruby V2](#)

Image

Provides the input image either as bytes or an S3 object.

You pass image bytes to an Amazon Rekognition API operation by using the `Bytes` property. For example, you would use the `Bytes` property to pass an image loaded from a local file system. Image bytes passed by using the `Bytes` property must be base64-encoded. Your code may not need to encode image bytes if you are using an AWS SDK to call Amazon Rekognition API operations.

For more information, see [로컬 파일 시스템에서 불러온 이미지 분석 \(p. 40\)](#).

You pass images stored in an S3 bucket to an Amazon Rekognition API operation by using the `S3Object` property. Images stored in an S3 bucket do not need to be base64-encoded.

The region for the S3 bucket containing the S3 object must match the region you use for Amazon Rekognition operations.

If you use the AWS CLI to call Amazon Rekognition operations, passing image bytes using the `Bytes` property is not supported. You must first upload the image to an Amazon S3 bucket and then call the operation using the `S3Object` property.

For Amazon Rekognition to process an S3 object, the user must have permission to access the S3 object. For more information, see [리소스 기반 정책 \(p. 27\)](#).

Contents

Bytes

Blob of image bytes up to 5 MBs.

Type: Base64-encoded binary data object

Length Constraints: Minimum length of 1. Maximum length of 5242880.

Required: No

S3Object

Identifies an S3 object as the image source.

Type: [S3Object \(p. 387\)](#) object

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java](#)
- [AWS SDK for Ruby V2](#)

ImageQuality

Identifies face image brightness and sharpness.

Contents

Brightness

Value representing brightness of the face. The service returns a value between 0 and 100 (inclusive). A higher value indicates a brighter face image.

Type: Float

Required: No

Sharpness

Value representing sharpness of the face. The service returns a value between 0 and 100 (inclusive). A higher value indicates a sharper face image.

Type: Float

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java](#)
- [AWS SDK for Ruby V2](#)

Instance

An instance of a label detected by [DetectLabels \(p. 247\)](#).

Contents

BoundingBox

The position of the label instance on the image.

Type: [BoundingBox \(p. 344\)](#) object

Required: No

Confidence

The confidence that Amazon Rekognition Image has in the accuracy of the bounding box.

Type: Float

Valid Range: Minimum value of 0. Maximum value of 100.

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java](#)
- [AWS SDK for Ruby V2](#)

KinesisDataStream

The Kinesis data stream Amazon Rekognition to which the analysis results of a Amazon Rekognition stream processor are streamed. For more information, see [CreateStreamProcessor \(p. 227\)](#).

Contents

Arn

ARN of the output Amazon Kinesis Data Streams stream.

Type: String

Pattern: (^arn:([a-z\d-]+):kinesis:([a-z\d-]+):\d{12}:+\$)

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java](#)
- [AWS SDK for Ruby V2](#)

KinesisVideoStream

Kinesis video stream stream that provides the source streaming video for a Amazon Rekognition Video stream processor. For more information, see [CreateStreamProcessor \(p. 227\)](#).

Contents

Arn

ARN of the Kinesis video stream stream that streams the source video.

Type: String

Pattern: (^arn:([a-z\d-]+):kinesisvideo:([a-z\d-]+):\d{12}:+\$)

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java](#)
- [AWS SDK for Ruby V2](#)

Label

Structure containing details about the detected label, including the name, and level of confidence.

The Amazon Rekognition Image operation [DetectLabels \(p. 247\)](#) operation returns a hierarchical taxonomy (`Parents`) for detected labels and also bounding box information (`Instances`) for detected labels. Amazon Rekognition Video doesn't return this information and [GetLabelDetection \(p. 278\)](#) returns `null` for the `Parents` and `Instances` attributes.

Contents

Confidence

Level of confidence.

Type: Float

Valid Range: Minimum value of 0. Maximum value of 100.

Required: No

Instances

If `Label` represents an object, `Instances` contains the bounding boxes for each instance of the detected object. Bounding boxes are returned for common object labels such as people, cars, furniture, apparel or pets.

Note

Amazon Rekognition Video does not support bounding box information for detected labels. The value of `Instances` is returned as `null` by [GetLabelDetection](#).

Type: Array of [Instance \(p. 370\)](#) objects

Required: No

Name

The name (label) of the object or scene.

Type: String

Required: No

Parents

The parent labels for a label. The response includes all ancestor labels.

Note

Amazon Rekognition Video does not support a hierarchical taxonomy of detected labels. The value of `Parents` is returned as `null` by [GetLabelDetection](#).

Type: Array of [Parent \(p. 381\)](#) objects

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for Ruby V2

LabelDetection

Information about a label detected in a video analysis request and the time the label was detected in the video.

Contents

Label

Details about the detected label.

Type: [Label \(p. 373\)](#) object

Required: No

Timestamp

Time, in milliseconds from the start of the video, that the label was detected.

Type: Long

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java](#)
- [AWS SDK for Ruby V2](#)

Landmark

Indicates the location of the landmark on the face.

Contents

Type

Type of landmark.

Type: String

Valid Values: 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

Required: No

X

The x-coordinate from the top left of the landmark expressed as the ratio of the width of the image. For example, if the image is 700 x 200 and the x-coordinate of the landmark is at 350 pixels, this value is 0.5.

Type: Float

Required: No

Y

The y-coordinate from the top left of the landmark expressed as the ratio of the height of the image. For example, if the image is 700 x 200 and the y-coordinate of the landmark is at 100 pixels, this value is 0.5.

Type: Float

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java](#)
- [AWS SDK for Ruby V2](#)

ModerationLabel

Provides information about a single type of moderated content found in an image or video. Each type of moderated content has a label within a hierarchical taxonomy. For more information, see [비안전 콘텐츠 감지](#) (p. 187).

Contents

Confidence

Specifies the confidence that Amazon Rekognition has that the label has been correctly identified.

If you don't specify the `MinConfidence` parameter in the call to `DetectModerationLabels`, the operation returns labels with a confidence value greater than or equal to 50 percent.

Type: Float

Valid Range: Minimum value of 0. Maximum value of 100.

Required: No

Name

The label name for the type of content detected in the image.

Type: String

Required: No

ParentName

The name for the parent label. Labels at the top level of the hierarchy have the parent label " ".

Type: String

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java](#)
- [AWS SDK for Ruby V2](#)

MouthOpen

Indicates whether or not the mouth on the face is open, and the confidence level in the determination.

Contents

Confidence

Level of confidence in the determination.

Type: Float

Valid Range: Minimum value of 0. Maximum value of 100.

Required: No

Value

Boolean value that indicates whether the mouth on the face is open or not.

Type: Boolean

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java](#)
- [AWS SDK for Ruby V2](#)

Mustache

Indicates whether or not the face has a mustache, and the confidence level in the determination.

Contents

Confidence

Level of confidence in the determination.

Type: Float

Valid Range: Minimum value of 0. Maximum value of 100.

Required: No

Value

Boolean value that indicates whether the face has mustache or not.

Type: Boolean

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java](#)
- [AWS SDK for Ruby V2](#)

NotificationChannel

The Amazon Simple Notification Service topic to which Amazon Rekognition publishes the completion status of a video analysis operation. For more information, see [Amazon Rekognition Video 작업 블러오기 \(p. 59\)](#).

Contents

RoleArn

The ARN of an IAM role that gives Amazon Rekognition publishing permissions to the Amazon SNS topic.

Type: String

Pattern: `arn:aws:iam::\d{12}:role/?[a-zA-Z_0-9+=,.@-_/.]+`

Required: Yes

SNSTopicArn

The Amazon SNS topic to which Amazon Rekognition posts the completion status.

Type: String

Pattern: `(^arn:aws:sns:.*:\w{12}:+$)`

Required: Yes

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java](#)
- [AWS SDK for Ruby V2](#)

Parent

A parent label for a label. A label can have 0, 1, or more parents.

Contents

Name

The name of the parent label.

Type: String

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java](#)
- [AWS SDK for Ruby V2](#)

PersonDetail

Details about a person detected in a video analysis request.

Contents

BoundingBox

Bounding box around the detected person.

Type: [BoundingBox \(p. 344\)](#) object

Required: No

Face

Face details for the detected person.

Type: [FaceDetail \(p. 359\)](#) object

Required: No

Index

Identifier for the person detected person within a video. Use to keep track of the person throughout the video. The identifier is not stored by Amazon Rekognition.

Type: Long

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java](#)
- [AWS SDK for Ruby V2](#)

PersonDetection

Details and path tracking information for a single time a person's path is tracked in a video. Amazon Rekognition operations that track people's paths return an array of `PersonDetection` objects with elements for each time a person's path is tracked in a video.

For more information, see [GetPersonTracking \(p. 282\)](#).

Contents

Person

Details about a person whose path was tracked in a video.

Type: [PersonDetail \(p. 382\)](#) object

Required: No

Timestamp

The time, in milliseconds from the start of the video, that the person's path was tracked.

Type: Long

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java](#)
- [AWS SDK for Ruby V2](#)

PersonMatch

Information about a person whose face matches a face(s) in an Amazon Rekognition collection. Includes information about the faces in the Amazon Rekognition collection ([FaceMatch \(p. 363\)](#)), information about the person ([PersonDetail \(p. 382\)](#)), and the time stamp for when the person was detected in a video. An array of `PersonMatch` objects is returned by [GetFaceSearch \(p. 273\)](#).

Contents

FaceMatches

Information about the faces in the input collection that match the face of a person in the video.

Type: Array of [FaceMatch \(p. 363\)](#) objects

Required: No

Person

Information about the matched person.

Type: [PersonDetail \(p. 382\)](#) object

Required: No

Timestamp

The time, in milliseconds from the beginning of the video, that the person was matched in the video.

Type: Long

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java](#)
- [AWS SDK for Ruby V2](#)

Point

The X and Y coordinates of a point on an image. The X and Y values returned are ratios of the overall image size. For example, if the input image is 700x200 and the operation returns X=0.5 and Y=0.25, then the point is at the (350,50) pixel coordinate on the image.

An array of `Point` objects, `Polygon`, is returned by [DetectText \(p. 254\)](#). `Polygon` represents a fine-grained polygon around detected text. For more information, see [Geometry \(p. 367\)](#).

Contents

X

The value of the X coordinate for a point on a `Polygon`.

Type: `Float`

Required: No

Y

The value of the Y coordinate for a point on a `Polygon`.

Type: `Float`

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java](#)
- [AWS SDK for Ruby V2](#)

Pose

Indicates the pose of the face as determined by its pitch, roll, and yaw.

Contents

Pitch

Value representing the face rotation on the pitch axis.

Type: Float

Valid Range: Minimum value of -180. Maximum value of 180.

Required: No

Roll

Value representing the face rotation on the roll axis.

Type: Float

Valid Range: Minimum value of -180. Maximum value of 180.

Required: No

Yaw

Value representing the face rotation on the yaw axis.

Type: Float

Valid Range: Minimum value of -180. Maximum value of 180.

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java](#)
- [AWS SDK for Ruby V2](#)

S3Object

Provides the S3 bucket name and object name.

The region for the S3 bucket containing the S3 object must match the region you use for Amazon Rekognition operations.

For Amazon Rekognition to process an S3 object, the user must have permission to access the S3 object. For more information, see [리소스 기반 정책 \(p. 27\)](#).

Contents

Bucket

Name of the S3 bucket.

Type: String

Length Constraints: Minimum length of 3. Maximum length of 255.

Pattern: [0-9A-Za-z\.\-_]*

Required: No

Name

S3 object key name.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1024.

Required: No

Version

If the bucket is versioning enabled, you can specify the object version.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1024.

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java](#)
- [AWS SDK for Ruby V2](#)

Smile

Indicates whether or not the face is smiling, and the confidence level in the determination.

Contents

Confidence

Level of confidence in the determination.

Type: Float

Valid Range: Minimum value of 0. Maximum value of 100.

Required: No

Value

Boolean value that indicates whether the face is smiling or not.

Type: Boolean

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java](#)
- [AWS SDK for Ruby V2](#)

StreamProcessor

An object that recognizes faces in a streaming video. An Amazon Rekognition stream processor is created by a call to [CreateStreamProcessor \(p. 227\)](#). The request parameters for `CreateStreamProcessor` describe the Kinesis video stream source for the streaming video, face recognition parameters, and where to stream the analysis results.

Contents

Name

Name of the Amazon Rekognition stream processor.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 128.

Pattern: [a-zA-Z0-9_.\-\-]+

Required: No

Status

Current status of the Amazon Rekognition stream processor.

Type: String

Valid Values: STOPPED | STARTING | RUNNING | FAILED | STOPPING

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java](#)
- [AWS SDK for Ruby V2](#)

StreamProcessorInput

Information about the source streaming video.

Contents

KinesisVideoStream

The Kinesis video stream input stream for the source streaming video.

Type: [KinesisVideoStream \(p. 372\)](#) object

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java](#)
- [AWS SDK for Ruby V2](#)

StreamProcessorOutput

Information about the Amazon Kinesis Data Streams stream to which a Amazon Rekognition Video stream processor streams the results of a video analysis. For more information, see [CreateStreamProcessor \(p. 227\)](#).

Contents

KinesisDataStream

The Amazon Kinesis Data Streams stream to which the Amazon Rekognition stream processor streams the analysis results.

Type: [KinesisDataStream \(p. 371\)](#) object

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java](#)
- [AWS SDK for Ruby V2](#)

StreamProcessorSettings

Input parameters used to recognize faces in a streaming video analyzed by a Amazon Rekognition stream processor.

Contents

FaceSearch

Face search settings to use on a streaming video.

Type: [FaceSearchSettings \(p. 365\)](#) object

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java](#)
- [AWS SDK for Ruby V2](#)

Sunglasses

Indicates whether or not the face is wearing sunglasses, and the confidence level in the determination.

Contents

Confidence

Level of confidence in the determination.

Type: Float

Valid Range: Minimum value of 0. Maximum value of 100.

Required: No

Value

Boolean value that indicates whether the face is wearing sunglasses or not.

Type: Boolean

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java](#)
- [AWS SDK for Ruby V2](#)

TextDetection

Information about a word or line of text detected by [DetectText \(p. 254\)](#).

The `DetectedText` field contains the text that Amazon Rekognition detected in the image.

Every word and line has an identifier (`Id`). Each word belongs to a line and has a parent identifier (`ParentId`) that identifies the line of text in which the word appears. The word `Id` is also an index for the word within a line of words.

For more information, see [텍스트 감지 \(p. 196\)](#).

Contents

Confidence

The confidence that Amazon Rekognition has in the accuracy of the detected text and the accuracy of the geometry points around the detected text.

Type: Float

Valid Range: Minimum value of 0. Maximum value of 100.

Required: No

DetectedText

The word or line of text recognized by Amazon Rekognition.

Type: String

Required: No

Geometry

The location of the detected text on the image. Includes an axis aligned coarse bounding box surrounding the text and a finer grain polygon for more accurate spatial information.

Type: [Geometry \(p. 367\)](#) object

Required: No

Id

The identifier for the detected text. The identifier is only unique for a single call to `DetectText`.

Type: Integer

Valid Range: Minimum value of 0.

Required: No

ParentId

The Parent identifier for the detected text identified by the value of `ID`. If the type of detected text is `LINE`, the value of `ParentId` is `Null`.

Type: Integer

Valid Range: Minimum value of 0.

Required: No

Type

The type of text that was detected.

Type: String

Valid Values: LINE | WORD

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java](#)
- [AWS SDK for Ruby V2](#)

UnindexedFace

A face that [IndexFaces \(p. 287\)](#) detected, but didn't index. Use the `Reasons` response attribute to determine why a face wasn't indexed.

Contents

FaceDetail

The structure that contains attributes of a face that `IndexFaces` detected, but didn't index.

Type: [FaceDetail \(p. 359\)](#) object

Required: No

Reasons

An array of reasons that specify why a face wasn't indexed.

- `EXTREME_POSE` - The face is at a pose that can't be detected. For example, the head is turned too far away from the camera.
- `EXCEEDS_MAX_FACES` - The number of faces detected is already higher than that specified by the `MaxFaces` input parameter for `IndexFaces`.
- `LOW_BRIGHTNESS` - The image is too dark.
- `LOW_SHARPNESS` - The image is too blurry.
- `LOW_CONFIDENCE` - The face was detected with a low confidence.
- `SMALL_BOUNDING_BOX` - The bounding box around the face is too small.

Type: Array of strings

Valid Values: `EXCEEDS_MAX_FACES` | `EXTREME_POSE` | `LOW_BRIGHTNESS` | `LOW_SHARPNESS` | `LOW_CONFIDENCE` | `SMALL_BOUNDING_BOX`

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java](#)
- [AWS SDK for Ruby V2](#)

Video

Video file stored in an Amazon S3 bucket. Amazon Rekognition video start operations such as [StartLabelDetection \(p. 330\)](#) use Video to specify a video for analysis. The supported file formats are .mp4, .mov and .avi.

Contents

S3Object

The Amazon S3 bucket name and file name for the video.

Type: [S3Object \(p. 387\)](#) object

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java](#)
- [AWS SDK for Ruby V2](#)

VideoMetadata

Information about a video that Amazon Rekognition analyzed. Videometadata is returned in every page of paginated responses from a Amazon Rekognition video operation.

Contents

Codec

Type of compression used in the analyzed video.

Type: String

Required: No

DurationMillis

Length of the video in milliseconds.

Type: Long

Valid Range: Minimum value of 0.

Required: No

Format

Format of the analyzed video. Possible values are MP4, MOV and AVI.

Type: String

Required: No

FrameHeight

Vertical pixel dimension of the video.

Type: Long

Valid Range: Minimum value of 0.

Required: No

FrameRate

Number of frames per second in the video.

Type: Float

Required: No

FrameWidth

Horizontal pixel dimension of the video.

Type: Long

Valid Range: Minimum value of 0.

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for Ruby V2

Amazon Rekognition의 제한

다음은 Amazon Rekognition에서의 제한 목록입니다. 변경 가능한 제한에 대한 자세한 내용은 [AWS 서비스 제한](#)을 참조하십시오. 제한을 변경하려면 [Create Case](#) 단원을 참조하십시오.

Amazon Rekognition Image

- Amazon S3 객체로 저장되는 이미지 최대 크기는 15MB로 제한됩니다.
- 높이 및 너비의 최소 픽셀 해상도는 80픽셀입니다.
- 감지하려면 얼굴이 1920x1080픽셀의 이미지에서 40x40픽셀보다 작아서는 안 됩니다. 1920X1080픽셀보다 높은 차원의 이미지는 비례적으로 더 큰 최소 얼굴 크기가 필요합니다.
- 파라미터로 API에 전달되는 원시 바이트의 최대 이미지 크기는 5MB입니다.
- Amazon Rekognition은 PNG 및 JPEG 이미지 형식을 지원합니다. 즉, DetectLabels 및 IndexFaces 등 다양한 API 작업에 입력으로 제공하는 이미지는 지원되는 다음 형식 중 하나여야 합니다.
- 단일 얼굴 모음에 저장할 수 있는 최대 얼굴 수는 20만 개입니다.
- 검색 API가 반환하는 일치 얼굴의 최대 수는 4096개입니다.
- 이미지 작업에 대한 계정당 기본 TPS(초당 트랜잭션) 제한은 다음과 같습니다.

이미지 작업	기본 TPS 제한
이미지 data plane 작업: <ul style="list-style-type: none">the section called “CompareFaces” (p. 219)the section called “DetectFaces” (p. 243)the section called “DetectLabels” (p. 247)the section called “DetectModerationLabels” (p. 251)the section called “DetectText” (p. 254)the section called “GetCelebrityInfo” (p. 257)the section called “IndexFaces” (p. 287)the section called “ListFaces” (p. 298)the section called “RecognizeCelebrities” (p. 304)the section called “SearchFaces” (p. 308)the section called “SearchFacesByImage” (p. 311)	<ul style="list-style-type: none">미국 동부(버지니아 북부) 지역 – 50미국 서부(오레곤) 지역 – 50EU(아일랜드) 지역 – 50미국 동부(오하이오) 리전 – 5아시아 태평양(시드니) 리전 – 5아시아 태평양(도쿄) 리전 – 5AWS GovCloud (US) – 5
이미지 control plane 작업: <ul style="list-style-type: none">the section called “CreateCollection” (p. 224)the section called “DeleteCollection” (p. 230)the section called “DeleteFaces” (p. 232)the section called “ListCollections” (p. 295)	Amazon Rekognition가 지원하는 각 리전에서 – 5

Amazon Rekognition Video 저장된 비디오

- Amazon Rekognition Video는 최대 8GB의 저장된 비디오를 분석할 수 있습니다.
- Amazon Rekognition Video는 계정당 최대 20개의 동시 작업을 지원합니다.

- 비디오 스토리지 작업에 대한 계정당 기본 TPS 제한은 다음과 같습니다.

비디오 스토리지 작업	기본 TPS 제한
저장된 비디오 Start 작업: <ul style="list-style-type: none">the section called "StartCelebrityRecognition" (p. 315)the section called "StartContentModeration" (p. 318)the section called "StartFaceDetection" (p. 322)the section called "StartFaceSearch" (p. 326)the section called "StartLabelDetection" (p. 330)the section called "StartPersonTracking" (p. 334)	Rekognition가 지원하는 각 리전에서 – 5
저장된 비디오 Get 작업: <ul style="list-style-type: none">the section called "GetCelebrityRecognition" (p. 259)the section called "GetContentModeration" (p. 264)the section called "GetFaceDetection" (p. 268)the section called "GetFaceSearch" (p. 273)the section called "GetLabelDetection" (p. 278)the section called "GetPersonTracking" (p. 282)	<ul style="list-style-type: none">미국 동부(버지니아 북부) 지역 – 20미국 서부(오레곤) 지역 – 20EU(아일랜드) 지역 – 20미국 동부(오하이오) 리전 – 5아시아 태평양(시드니) 리전 – 5아시아 태평양(도쿄) 리전 – 5AWS GovCloud (US) – 5

Amazon Rekognition Video 스트리밍 비디오

- Kinesis Video 입력 스트림은 최대 1개의 Amazon Rekognition Video 스트림 프로세서와 연결될 수 있습니다.
- Kinesis Data 출력 스트림은 최대 1개의 Amazon Rekognition Video 스트림 프로세서와 연결될 수 있습니다.
- Amazon Rekognition Video 스트림 프로세서와 연결된 Kinesis Video 입력 스트림 및 Kinesis Data 출력 스트림은 여러 프로세서와 공유될 수 없습니다.
- 특정 AWS 계정의 경우 단일 리전에 동시 존재할 수 있는 Amazon Rekognition Video 스트림 프로세서의 기본 수는 10개입니다.
- 모든 스트리밍 비디오 스토리지 작업에 대한 계정당 기본 TPS 제한은 다음과 같습니다.

비디오 스트리밍 작업	기본 TPS 제한
<ul style="list-style-type: none">the section called "CreateStreamProcessor" (p. 227)the section called "DeleteStreamProcessor" (p. 234)the section called "DescribeStreamProcessor" (p. 239)the section called "ListStreamProcessors" (p. 301)the section called "StartStreamProcessor" (p. 337)the section called "StopStreamProcessor" (p. 339)	Amazon Rekognition가 지원하는 각 리전에서 – 1

Amazon Rekognition 문서 이력

다음 표에서는 Amazon Rekognition 개발자 안내서의 각 릴리스에서 변경된 중요 사항에 대해 설명합니다. 이 설명서에 대한 업데이트 알림을 받으려면 RSS 피드를 구독하면 됩니다.

- 최종 설명서 업데이트: 2018년 5월 29일

update-history-change	update-history-description	update-history-date
새 Python 예제 (p. 402)	Python 예제가 Amazon Rekognition Video 콘텐츠에 추가되고 일부 콘텐츠가 재구성되었습니다.	June 26, 2018
콘텐츠 레이아웃 업데이트 (p. 402)	Amazon Rekognition Image 콘텐츠가 재구성되고 새 Python 및 C# 예제가 추가되었습니다.	May 29, 2018
Amazon Rekognition은 AWS CloudTrail 지원 (p. 402)	Amazon Rekognition은 사용자, 역할 또는 Amazon Rekognition에서 AWS 서비스가 수행한 작업 목록을 알려 주는 AWS CloudTrail 서비스와 통합됩니다. 자세한 내용은 Logging Amazon Rekognition API Calls with AWS CloudTrail 단원을 참조하십시오.	April 6, 2018
저장된 비디오와 스트리밍 비디오를 분석합니다. 새로운 목차 (p. 402)	저장된 비디오 분석에 대한 자세한 내용은 저장된 비디오 작업 을 참조하십시오. 스트리밍 비디오 분석에 대한 자세한 내용은 스트리밍 비디오 작업 을 참조하십시오. Amazon Rekognition 문서의 목차가 이미지와 비디오 작업에 대한 내용을 반영해 재구성되었습니다.	November 29, 2017
이미지 및 얼굴 인식 모델의 텍스트 (p. 402)	이제 Amazon Rekognition이 이미지에서 텍스트를 감지할 수 있습니다. 자세한 내용은 텍스트 감지 를 참조하십시오. Amazon Rekognition의 얼굴 인식 딥 러닝 모델 버전 관리를 소개합니다. 자세한 내용은 모델 버전 관리 를 참조하십시오.	November 21, 2017
유명 인사 인식 (p. 402)	이제 Amazon Rekognition이 유명 인사 이미지를 분석할 수 있습니다. 자세한 내용은 유명 인사 인식 을 참조하십시오.	June 8, 2017
이미지 조정 (p. 402)	Amazon Rekognition은 이제 이미지에 노골적이거나 선정적인 성인 콘텐츠가 포함되어 있는지 판단할	April 19, 2017

수 있습니다. 자세한 내용은 [비안전 콘텐츠 감지](#)를 참조하십시오.

감지된 얼굴의 연령 범위.
[Aggregated Rekognition Metrics 장](#) (p. 402)

Amazon Rekognition은 이제 Rekognition API가 감지한 얼굴의 추정 연령 범위(단위: 세)를 반환합니다. 자세한 내용은 [AgeRange](#)를 참조하십시오. 이제 Rekognition 콘솔에는 지정된 기간 동안 Rekognition의 Amazon CloudWatch 측정치 집계를 위한 활동 그래프를 보여 주는 측정치 창이 있습니다. 자세한 내용은 [연습 4: 집계 측정치 보기\(콘솔\)](#)를 참조하십시오.

February 9, 2017

새로운 서비스 및 가이드 (p. 402)

이것은 이미지 분석 서비스인 Amazon Rekognition과 Amazon Rekognition 개발자 가이드의 최초 릴리스입니다.

November 30, 2016

AWS Glossary

For the latest AWS terminology, see the [AWS Glossary](#) in the AWS General Reference.